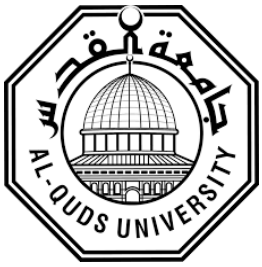
Deanship of Graduate Studies

Al-Quds University



Data Injection in ICMP Protocol Vulnerability and Exploitation

Eyad Fatih Ali Refai

M.Sc. Thesis

Jerusalem – Palestine

1442 / 2020

Data Injection in ICMP Protocol Vulnerability and Exploitation

Prepared by:

Eyad Fatih Ali Refai

Advisor: Dr. Raid Zaghal

Co-Advisor: Dr. **Saeed Salah**

"A thesis submitted to the Faculty of Science and Technology, Al-Quds University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer"

1442 / 2020

Al-Quds University

Deanship of Graduate Studies

Computer Science

Thesis Approval

Data Injection in ICMP Protocol Vulnerability and Exploitation

Prepared by: Eyad Fatih Ali Refai

Registration No.: 21712293

Advisor: Dr. Raid Zaghal

Co-Advisor: Dr. Saeed Salah

Master Thesis submitted and accepted, Date: / / 2020

The names and signatures of the examining committee members are as follows:

1- Head of Committee: Dr. Raid Zaghal Signature ……

2- Co-Advisor: Dr. Saeed Salah Signature …...

3- Internal Examiner: Dr. Signature ……

4- External Examiner: Dr. Signature ……

Jerusalem – Palestine

1442 / 2020

Dedication:

To my parents who are still caring for me as I was a child, to my wife who always supports me, to my daughters, the source of my smile, to my brothers and sisters, to all of them I dedicate this thesis work.

Eyad Fatih Ali Refai

Declaration:

I certify that this thesis work submitted for the degree of Master in Computer Science results from my research, except where otherwise acknowledged, and that this thesis (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed: ----------

Eyad Fatih Ali Refai

Date: / /2020.

Acknowledgments:

I would like to express my thanks and gratitude to my Thesis advisor Dr. Raid Zaghal and co-advisor, Dr. Saeed Salah, who encouraged, motivated, and taught me patience to complete this thesis work based on proper scientific research methods.

Also, I would like to thank the Department of Computer Science graduate studies committee who supported me and treated me with kindness and friendliness.

Eyad Fatih Ali Refai

Abstract

This thesis and scientific research aim to prove and review the possibility of injecting the ICMP protocol packets with encrypted or unencrypted data and passing it to another network, external or internal receiver. We will also see the potential and ability of modern generations in the network structure to follow up, monitor, or block such incorrect behavior due to this protocol's inefficient use.

As ICMP is one of the critical protocols at work, this protocol's exploitation began to threaten networks and their components and enable their work and privacy. Cyber-attacks or electronic attacks in various forms are an old/new phenomenon that the world is experiencing. Still, it started to increase due to the considerable expansion of networks' usage and their applications and systems, which poses a threat to these network administrators and employees. Different technologies have emerged to exploit the structure of building these networks to use their software components to carry out cyber-attacks. It affects systems such as blocking the service, disabling access, or electronic spying on packets and other forms.

Our study problem has emerged after research and audit and through the work of the author at the Center for Combating Cyber Crime in Palestine. The need to combat this phenomenon is an essential part of understanding networks' general structure and how they and their components work. Thus, our study emerged in relation to leaking data and injecting data within a legal packet. We will perform several real experiments to prove our concept and then draw recommendations and conclusions.

We will work on technical experiments to prove the validity of scientific research. It has already appeared to us that data can be injected into the packages, which may cause data to leak, and modification in the header of the packet. Also, this type of change shows the emergence of a

new kind of cyber-attacks that may be caused by the ICMP package as many from cyber-attacks using these packages.

حقن البيانات في نقاط ضعف بروتوكول ICMP واستغلاله

اعداد: اياد فتيح علي الرفاعي

المشرف الاول: د. رائد الزغل

المشرف الثاني: د. سعيد صلاح

الملخص

تهدف هذه الرسالة والبحث العلمي الى اثبات واستعراض امكانية حقن حزمة البروتوكول ICMP ببيانات مشفره او غير مشفرة وتمريرها الى شبكة اخرى او مستقبل خارجي او داخلي، بالإضافة الى ذلك سنرى امكانية وقدرة الاجيال الحديثة في هيكلية الشبكات لمتابعة او رصد او حجب مثل هذا النوع من السلوك الغير صحيح للاستخدام الغير أمثل لهذا البروتوكول. حيث يعتبر بروتوكول ICMP من البروتوكولات ذات الاهمية في العمل وقد بدأ استغلال هذا البروتوكول في تشكيل تهديد وخطر على الشبكات ومكوناتها واتاحة عملها وخصوصيتها ايضا. تعتبر ظاهرة الهجمات السيبرانية او الهجمات الالكترونية بأشكالها المختلفة ظاهرة جديدة قديمة يعيشها العالم ولكن بدأت بالازدياد نتيجة التوسع الكبير في استخدام الشبكات وتطبيقاتها وانظمتها، مما يشكل خطراً على مديري هذه الشبكات والعاملين فيها. وقد ظهرت تقنيات مختلفة لاستغلال هيكلية بناء هذه الشبكات من حيث استغلال مكوناتها البرمجية في ارتكاب هجمات سيبرانية تؤثر على عمل الشبكات مثل حجب الخدمة، أو تعطيل الوصول، أو التجسس الالكتروني على الحزم، وغيرها من أشكال الخروقات الأمنية.

إن تطور ادوات التقنية بشكلها الحالي قد يساهم ايضا في الوصول الى هجمات واستغلالها بشكل غير اخلاقي وسيؤثر ذلك على الصلاحيات الفنية الممنوحة للفنين في داخل اقسام العمل في مراكز البيانات المختلفة او الذين يعملون داخل اقسام ذات حساسية عالية ومعلومات حساسة. وسوف نجيب في هذه الدراسة عن العديد من التساؤلات والتي بطبيعتها ستخلق منهجية أمن المعلومات، بحيث أنها يمكن أن تضاف على لوائح وسياسات أمن المعلومات كقضايا جديدة ويكون لها وظائف جديده.

لقد ظهرت المشكلة التي تعالجها هذه الدراسة بعد البحث والتدقيق، ومن خلال عملي في مركز مكافحة الجرائم السيبرانية في فلسطين، حيث تعتبر ضرورة مكافحة هذه الظواهر جزء مهم من فهم الهيكلية العامة للشبكات وطريقه عملها ومكوناتها، ومن هنا ظهرت مشكلة الدراسة الخاصة بنا التي تتعلق بتسريب بيانات وحقن بيانات داخل حزمة مهمة في العمل اليومي لمهندسي الشبكات والفنيين.

ايضا ستطرح هذه الدراسة العديد من الاسئلة والمواضيع للبحث العلمي بحيث تطرح مسالة تطوير وانتاج برمجيات ومعدات تساهم في الحد او تتبع او حفظ السجلات والتقاط الحزم او مساعدة البحث الجنائي الالكتروني في حال تم الاستغلال او اجراءات التدقيق الدوري للأنظمة وسلامة مركز البيانات وسرية وخصوصية البيانات. ففي "الطب الشرعي" الالكتروني يمكن الاستفادة من الحزمة التي يتم التقاطها في تحليل سلوك الشبكة او البرمجيات ذات السلوك الضار، أو تمرير بيانات الى أطراف غير مخول لها للوصول الى المعلومات، أو الاتصالات العكسية، ومن ثم إيجاد حلول للمشاكل التي من الممكن أن تقع فيها الشبكات من خلال إحصاء كمية البيانات التي تمر عبر الشبكة مما يساعد في اكتشاف عمليات الدخول الغير مرخصة، واكتشاف الهجمات من نوع (DoS) و (DDoS)، وتحليل بروتوكولات الشبكة وتنقيحها.

سنعمل في هذا البحث على تجارب فنية لإثبات صحة البحث العلمي وقد ظهر لدينا بالفعل انه يمكن حقن بيانات داخل الحزم مما قد يسبب في تسريب البيانات ، والتعديل في ترويسة الحزمة ، وايضا يظهر هذا النوع من التعديل ظهور نوع جديد من الهجمات السيبرانية التي قد تتسبب بها حزمة ICMP كما العديد من الهجمات السيبرانية بإستخدام هذه الحزم .

Table of Contents

List of Tables

List of Figures

List of Abbreviations used

ICMP- Internet Control Message Protocol

QoS - Quality of Service

IP - Internet Protocol

ID - Identity

TCP - Transmission Control Protocol

ASCII - American Standard Code for Information Interchange

OSI - Open System Interconnection

MAC - Media Access Control

UDP - User Datagram Protocol

OSPF - Open Shortest Path First

BGP - Border Gateway Protocol

SLIP - Serial Linux Internet Protocol

HDLC - High-Level Data Link Control

NFS - Network File System

HTTP - Hypertext Transfer Protocol

TCP - Transmission Control Protocol/

UDP - User Datagram Protocol

MTU - Maximum transmission unit

HTML - Hypertext Markup Language

XML - Extensible Markup Language

CSMA - Carrier-sense multiple access

TTL - Time to Live

DDOS - Distributed Denial-of-Service Attack

LAN - local area network

PCAP - Packet Capture

CSV - comma-separated values

GPSD - General Protocol Scanning Detection

Chapter 1

Introduction

Chapter 1 : Introduction

1.1. Introduction

Network and system workers often need to detect and control errors in their network infrastructure, ensure the QoS provided in the network, and communicate with the outside world at the appropriate ideal time. The best way to do this is to use the ICMP protocol, responsible for these functions. Still, this reliable protocol has become the biggest ambition for some pirates and saboteurs in exploiting them in a manner that achieves their goals in piracy or leaking data or information from within the network to the outside, or vice versa. As for structures that reduce such attacks, it increases the system's vulnerability, as the exploitation of what is reliable has become more ambitious. It is known that ICMP protocol has been used to launch several types of attacks including; DDOS attacks to disrupt systems in addition to ICMP attacks, route attacks, Smurf attacks, ping sweep attacks, and TCP Scan. However, the tremendous development of analysis and tools contributing to reducing these attacks negatively impacted these systems' quality, efficiency, and performance. Specific solutions that have been proposed to regulate networks have become venues of attacks, too.

Some ICMP messages have unique structures and special headers that are different from other messages. Specific protocol titles can be exploited and injected with confidential data, as in some cases, as will be shown in Chapter 2.

The detection of the packets and the analysis of the damage and the distinction became complicated as the behavior of the packet has undergone colossal development. The detection of errors and manipulation and the ability of attackers in manipulation are also increasing.

The spread of cybercrime culture and the search for gaps to exploit the weaknesses to violate the privacy of data or information and search in depth the mechanism of circumvention systems not to find or create technical evidence that contributes to tracking, accounting, or limiting similar behavior.

In this work, one ICMP header will be used to inject text data in the encrypted and confidential form through a valid and authentic request within one of the protocol headers. The message will also be represented in ASCII format and injected into one of the protocol headers using the ASCII system.

1.2. Problem Statement

This study's focus is to assess the possibility of exploiting the third layer of the network, particularly the potential exploitation of the ICMP, which will require changes in its protocol header.

The system will search practically in a set of secret data injection (encrypted or unencrypted) and send it hidden over the network in the form of text, using the ICMP protocol. We will use one of the structures and one of the types after experiencing the best in terms of efficiency, reliability, credibility, and good behavior (encrypted or unencrypted), to be sent using a Python-based programming code that contributes to the construction of the same original header but with some modifications on one of the fields to inject the message into it. On the other hand, the recipient will also work on a Python-based software interface that contributes to the confidential information about whether the information is encrypted or unencrypted.

1.3. Hypothesis

The hypotheses of this study are based on the answers to the questions presented in the problem of the study, which will be three hypotheses as follows:

1.3.1. The first hypothesis

Internal control procedures, policies, and measures in network protection systems prevent or limit the protocol's incorrect behavior in its database structure. This hypothesis is divided into the following assumptions:

- Internal control procedures for network administrators are not effective in preventing electronic fraud using ICMP.

- The procedures, policies, and measures in the systems are not effective in preventing the injection or circumvention of the headers injected with data or information to leak or pass them without leaving any trace.

1.3.2. The second hypothesis

ICMP protocol is essential in terms of use and targets because of its negative impact on exploiting the injection or diversion or exploitation of non-optimal products.

1.3.3. The third hypothesis

The ICMP packet's origin analysis and comparison will determine that there is already a directory but not within the encoded data because the encrypted data is characterized as having a text, not a concept. However, in hiding the data, it does not have the effect of having the universe's text very natural.

1.3 Methodology

This research follows the scientific approach in the search for information.

The methodology of the study will be divided into theoretical and applied study:

- **Theoretical study:**

The theoretical study will briefly identify the network layers and some of the protocols used and transmit the IP/TCP protocols and the ICMP header. The transition to hidden channel technology within the Torres and the possibility of finding hidden channels within the ICMP header and ICMP packet will be presented in the second chapter of this study.

- **Applied study:**

This study will search using an ICMP header to inject text data in an encrypted and confidential form through a valid and authentic request within one of the protocol headers. The message will also be represented in ASCII format and injected into one of the protocol headers. This is important since the ASCII format is small in size, has high transmission efficiency, and can preserve later

transmission data within the network. The reason for using the ASCII system is that computers only store and process numbers. ASCII forms a numerical representation of letters, numbers, and some special control characters. This will be presented in Chapter 3 of this study.

1.4 Objectives and Aims

This study's main objective is to prove that there is a possibility of manipulating and injecting data with different images inside one of the ICMP headers. If that can be proven, several things must be followed in the policies, systems, and usage of this protocol's representative within the network and network administrators.

1.5 Strengths and weaknesses of the study

➢ Strength:

- This study will add a new type of cyber-attacks that exploit the ICMP.
- This study will show that this ICMP header can already be used and will add new evidence that methods, methodologies, and theories should be used to detect this exploitation.
- The study does not consider the information aspect only but looks at the practical aspect of proving the case of exploitation.

➢ Weaknesses:

- The study will not find the appropriate environment for applying the mechanism of investigation or examination of the packet after data injection. This environment needs laboratories specialized in the analysis of behavior without affecting the packet's validity calculations.
- The use of the ASCII encryption system is weak if there is a packet capture.

1.6 Thesis Structure

This thesis is divided into six chapters. They are listed below with a brief description of the content of each one of them.

Chapter 1 (Introduction): This chapter includes the main idea and question of the thesis and summarizes its hypotheses and contributions.

Chapter 2 (Research Design and Methods): This chapter will study the previous background of network science and network layers' components and their role in our study problem. We will also review the ICMP protocol, its layer, and the types of protocols.

Chapter 3 (Literature Review): In this chapter, we will review previous studies where many researchers are still trying to prove that the various protocol heads are difficult to exploit incorrectly and build protection and control systems for networks that try to find new technologies to keep pace with the rapid development and potential capabilities of hackers and saboteurs.

Chapter 4 (Methodology): This chapter outlined the experimental procedure employed and presents the experimental data.

Chapter 5 (Simulation Results and Discussion): A data masking system has been proposed within the ICMP packet header after looking and researching and conducting many technical experiments on the ICMP packet building structure and modifying it so that we can demonstrate the status of the problem and answer the research questions and hypotheses. A modification has been made in building the ICMP packet architecture to take advantage of the hidden channels inside to send the most considerable amount of data regardless of whether it is encrypted or not.

Chapter 6 (Conclusions and Future Work): the chapter will present the major conclusions drawn from the experimental work. It also includes specific suggestions for future work.

Chapter 2

Research Design and Methods

Chapter 2 : Research Design and Methods

The communication workflow in networks between two devices is through entering the required data and sending it by the network layers that start with the application layer. This data is transmitted and translated by passing on all the sending device layers, starting with the application layer and ending with the physical layer where the data has been converted. The bits are ready to be transferred over the wires after each layer adds private information to the data that it wishes to send. This process is called Encapsulation. When it reaches the receiving device, the data passes the OSI layers backward, starting from the physical layer and ending with the application layer in the process. It is called De-Encapsulation, and the resulting data is what the receiving user sees on his device.

2.1. Layers of OSI

The various significant layers of the OSI are depicted in Table 2.1. The following subsections describe these layers in some detail.

2.1.1. Layer 1: Physical

It is the layer responsible for transmitting data processed by the upper layers through the transmission medium.

Data that can be information such, as text, images, sounds, are represented by electrical impulses called voltage on copper wires or optical fiber pulses. The transmission is called encoding or modification and is performed using cables and connectors.

2.1.2. Layer 2: Data Link

This layer provides access to physical networking and transmission modes, enabling data to find its intended destination in the network.

It provides a reliable link to data on a physical interface using the MAC address, the physical address on the network card, and framing to organize or collect data bits and flow control. The MAC is used to select which computer will send its binary data from a computer group, sending it simultaneously.

After dividing the data into smaller parts called frames, add extra information to the header and the trailer containing control information to ensure that the frames are error-free.

2.1.3. Layer 3: Network Layer

It is responsible for addressing messages and translating logical addresses and names into physical addresses that the network understands.

The logical address may be an e-mail or Internet address in this format: 192.168.0.100

The physical address shall be as follows: 5A.01.60.8c.01.03.

This layer selects the most suitable path between the transmitter and the receiver, so the routers operate within this layer. It is worth mentioning that all the routers are working in this layer.

ICMP works as a protocol for study in the network layer and the Internet and is used by IP to perform different tasks. ICMP is a protocol that manages and sends messages to IP. ICMP messages are in the form of IP packets at specific intervals; ICMP uses dots to perform some of the functions that will be separated later.

2.1.4. Layer 4: Transport Layer

The transport layer defines end-to-end connections between source and destination applications. Transport services include the following primary services:

- Divide top application data into segments.
- Establish operations between sending and receiving devices.
- Ensure that the data's reliability and accuracy and send a message to the receiving device to receive data.
- Control the flow of this data.

Also, choose the best route to send that data.

The most critical protocols of Layer-4 are the TCP and UDP protocols. They use port numbers (or sockets) to track different conversations that simultaneously cross the network to pass information to the top layer.

This layer is where the TCP protocol works, and this layer is one of the layers most engineers are excited about because it has a switching device. It is worth mentioning that only the software was working on this layer before manufacturing these devices. Hence, we can understand why TCP/IP is pronounced as a single packet and usually starts with TCP because it is above the third layer. Data is scrambled to reach the target device for several reasons, including faulty paths, or the router may ignore the data if it is too busy. Such a sender is identified if the transmitter fails to receive a notification message with data access. Many robust Routing Protocols work here in this layer, including OSPF, BGP, which is activated directly above the IP protocol.

2.1.5. Layer 5: Session Layer

The session layer allows two applications to synchronize their connections and exchange data. This layer divides communication between two systems into dialogue units and offers maximum and minimum synchronization points during this connection. It allows two programs on two different computers to connect, use, and terminate the two computers.

The layer is also responsible for identifying devices and their names and issuing reports on the communications you make. This layer also performs some administrative tasks such as the order of messages sent according to the time of sending them and the duration of sending each message. The protocols operating within this layer include:

- The x-window system.
- AppleTalk Session Protocol (ASP).
- Session Control Protocol for Digital Network Architecture (DNA SCP).
- Network File System (NFS).

This layer also takes a sample of the last piece of data sent when the network stopped working so that the data would be sent when the network returns to work from the point at which the transmission stop.

This layer is critical in e-commerce. It helps preserve the privacy of information in electronic purchases. This is accomplished by avoiding the so-called load balancing so that the direct contact is only between the buyer's and the seller's servers.

2.1.6. Layer 6: Presentation

The display layer ensures that the application layer in another system will read the application layer's information from one system. If necessary, this layer can be translated into several different data formats. This layer also compresses the data to reduce the number of bits to be moved. To understand the principle better, let us say that we have two people speaking different languages. The only way to understand each other is by having someone else translate. The presentation layer serves as a translator for devices that need to communicate over the network.

2.1.7. Layer 7: Application Layer

The layer is the closest user layer directly controlled by the user and provides network services to user applications. It differs from other layers in that it does not provide services for any other layer, but only for applications outside the OSI model. Examples of such applications are the work papers, word processing programs, and the Fund's staff's programs in banks.

This layer supports several programs, including file transfer programs, database programs, and e-mail programs from the protocols that work in this Telnet layer, HTTP.

The last layer deals directly with the user and various applications such as Telnet, FTP, and mail (pop3 and SMTP).

Table 2.1. OSI Layers functional [1]

	Layers #	Layers Name	Layers functional
Layers	1.	Physical	Hubs, Repeaters, Cables, Optical Fiber, SONET/SDN, Coaxial Cable, Twisted Pair Cable, and Connectors
	2.	Data Link	- 802.11 (WLAN), Wi-Fi, WiMAX, ATM, Ethernet, Token Ring, Frame Relay, PPTP, L2TP, and ISDN
	3.	Network	IPv4, IPV6, IPX, OSPF, ICMP, IGMP, and ARP
	4.	Transport	TCP, SPX, and UDP
	5.	Session layer	Logical Ports 21, 22, 23, 80 etc.
	6.	Presentation layer	SSL, WEP, WPA, Kerberos
	7.	Application Layer	DHCP, DNS, FTP, HTTP, IMAP4, NNTP, POP3, SMTP, SNMP, SSH, TELNET, and NTP

2.2. Network Layer 3 ICMP

Is the layer responsible for the possibility of communication between the devices, whether on a local network or an external network? The functions of this layer are addressing and routing, and uses the network layer IP protocol for addressing and sending data, as it specifies each MAC address (IP address) through a process called Reverse Address Resolution Protocol (RARP). Thus, the MAC

address carries a unique IP address that contributes to the device's definition and becomes as if it has a name on the network.

This layer allows the devices to share information about network problems or failures if one is to occur. The protocol responsible for this is the ICMP protocol.

The IP protocol consists of a set of fields per packet, as shown in the following image (Figure 2.1):

2.3. What is the ICMP PROTOCOL?

ICMP is one of the necessary protocols primarily used by the network computers' operating systems to send error messages. As an example: a service request is not available or be the host or router

Can not communicate with them if the ICMP depends on the IP to carry out its functions, as it is an integral part of IP and differs in purpose from transport protocols such as TCP and UDP. Therefore, it is not used to send and receive data between systems, which is usually not directly used by the network applications.

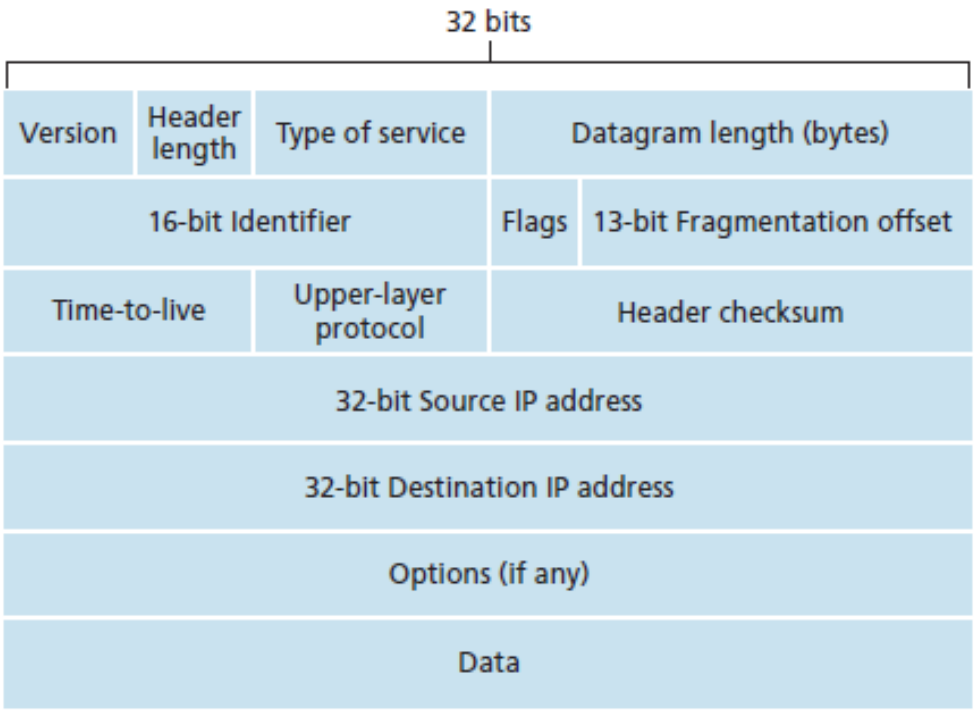


Figure 2.1. IP header format

ICMP is a support protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information that indicates, for example, that the requested service is unavailable or a host or router could not be reached. ICMP differs from transport protocols such as TCP and UDP. It is not commonly used for data exchange between systems, nor is it used regularly by end-user network applications (except for some diagnostic tools such as ping and traceroute testing).

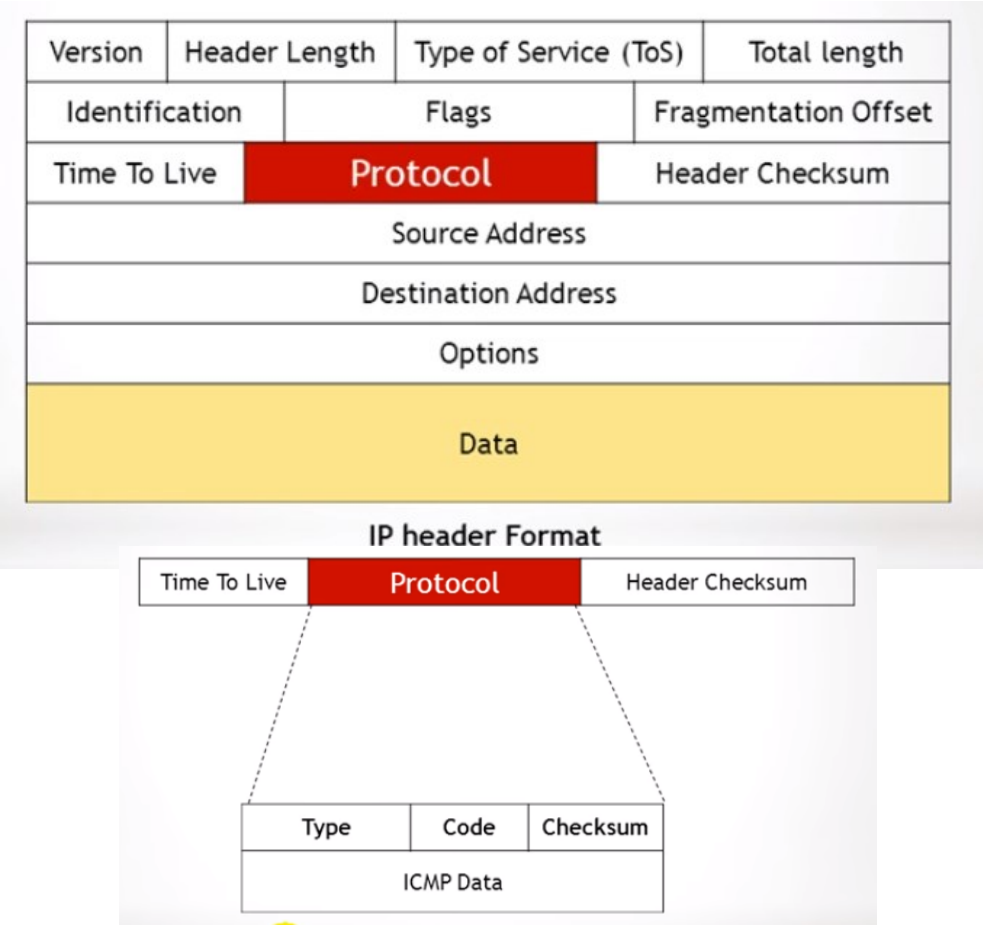


Figure 2.2. The picture shows the where about of ICMP packet

Many standard network utilities rely on ICMP messages (Figure 2.3). The traceroute command can be executed by sending IP datagrams with specially specified IP TTL header fields and finding ICMP time that is exceeded in inaccessible and destination messages that have generated a response. The relevant ping utility is performed by using an ICMP echo request and echo reply messages.

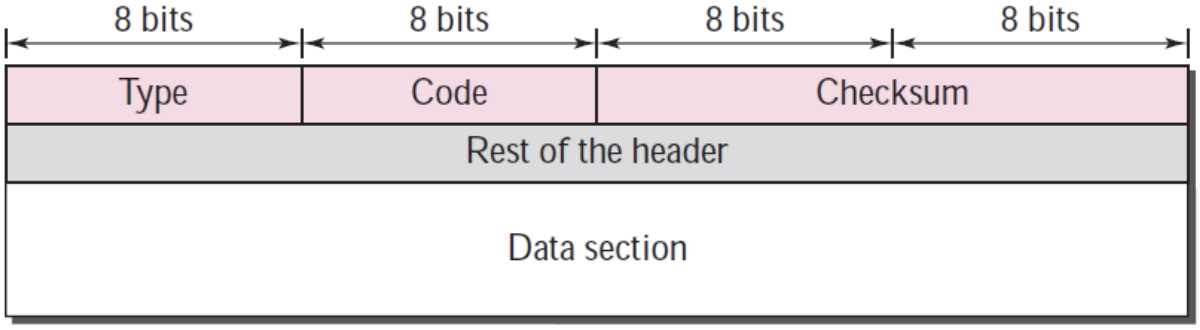


Figure 2.3. ICMP Message Format

ICMP uses necessary IP support as a high-level protocol; however, ICMP is already an integral part of the Internet protocol. Although ICMP messages are included in standard IP packets, ICMP messages are usually processed as a particular case and are distinguished from standard IP processing. In many cases, it is necessary to examine the ICMP message's contents and deliver the appropriate error message to the application responsible for sending the IP packet that sent the ICMP message.

ICMP is the network layer protocol. No TCP or UDP port number is bound to ICMP packets because these numbers are bound to the transport layer above[1].

2.4. ICMP Message Format

1. ICMP Type[2]

[1] https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model "The OSI Model's Seven Layers Defined and Functions Explained". *Microsoft Support*. Retrieved 2014-12-28.

[2] https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml#icmp-parameters-types

It shows us the type of message that is managed from this protocol and should be expressed in the header is a number, and each number has a name and meaning as in Table 2.2:

	Name	Type	Name
0	Echo Reply	20-29	Reserved (for Robustness Experiment)
1	Unassigned	30	
2	Unassigned	31	
3	Destination Unreachable	32	
4	Source Quench (Deprecated)	33	
5	Redirect	34	
6	Alternate Host Address (Deprecated)	35	
7	Unassigned	36	
8	Echo	37	
9	Router Advertisement	38	
10	Router Solicitation	39	
11	Time Exceeded	40	
12	Parameter Problem	41	
13	Timestamp	42	
14	Timestamp Reply	43	
15	Information Request (Deprecated)	44-252	
16	Information Reply (Deprecated)		
17	Address Mask Request (Deprecated)		

18	Address Mask Reply (Deprecated)		
19	Reserved (for Security)		

Table 2.2. ICMP type

2. ICMP Code [3]

In this table (Table 2.3), we have more instances of ICMP packet status.

Codes	Description	Codes	Description
0	Net Unreachable	11	Destination Network Unreachable for Type of Service
1	Host Unreachable	12	Destination Host Unreachable for Type of Service
2	Protocol Unreachable	13	Communication Administratively Prohibited
3	Port Unreachable	14	Host Precedence Violation
4	Fragmentation Needed and Don't Fragment was Set	15	Precedence cutoff in effect
5	Source Route Failed		
6	Destination Network Unknown		
7	Destination Host Unknown		
8	Source Host Isolated		
9	Communication with Destination The network is Administratively Prohibited		
10	Communication with Destination Host is Administratively Prohibited		

Table 2.3. ICMP code

[3] https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml

2.5 ICMP encapsulation

The ICMP packet is encapsulated in an IPv4 packet and consists of header and data sections:

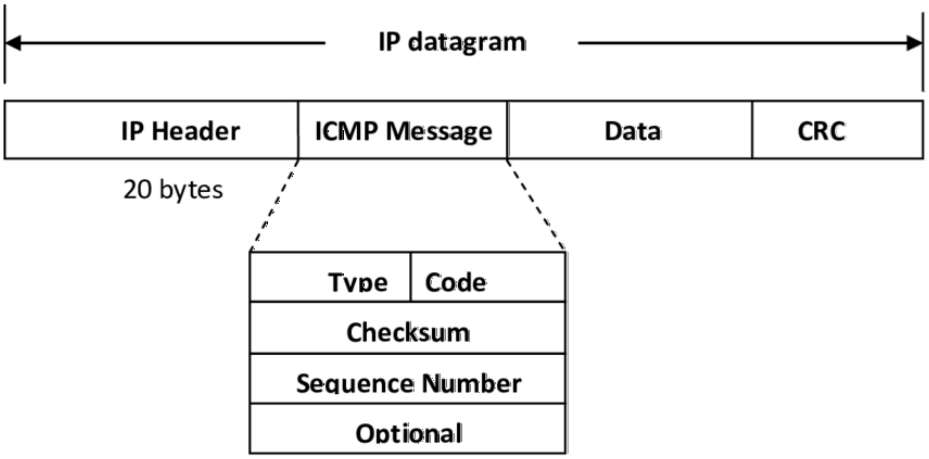


Figure 2.4. ICMP encapsulation

➢ Header

The ICMP header starts after the IPv4 header and is specified by a protocol number[4]. All ICMP packets contain an 8-byte header and a variable size data partition. The first four bytes of the header have a fixed format.[5]

Offsets	Octet	0	1	2	3
0	0	Type	Code	Checksum	
4	32	Rest of Header			

Table 2.4. ICMP header format

[4] *Forouzan, Behrouz A. (2007). Data Communications and Networking (Fourth Ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 978-0-07-296775-3.*

[5] Forouzan, Behrouz A. (2007). *Data Communications and Networking* (Fourth Ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 978-0-07-296775-3.

ICMP error messages contain a data partition that has a copy of the entire IPv4 header and at least the first eight bytes of the data from the IPv4 packet that caused the error message. The maximum length of ICMP error messages is 576 bytes.[6] This data is used by the host to match the message with the appropriate process. If the top-level protocol uses port numbers, it should be in the first eight bytes of the original datagram data.[7]

Our study object is located on Type 1 with ID +80 ping. The ICMP echo request to the target host. The target host responds with an echo response, which means that the target host is alive to take advantage of applying the data injection algorithm within the header within the original accounts and show this packet as valid after making these calculations.

Table 2.5. ICMP payload size

IP header	ICMP header	ICMP payload size	MTU (1500)
20 bytes	8 bytes	1472 bytes (maximum)	20 + 8 + 1472 = 1500

➢ Data Structures of ICMP

The three main data structures used by the ICMP code are:

- ICMP header "ICMPhdr". This will be the start of the function to build a header.
- ICMP control, and the ICMP message type descriptor. Among its routine fields used to process entry messages.
- icmp_bxm, the input structure is given as a parameter for the two send routines described in the "Send ICMP Messages" section. It includes all the information needed to transmit an ICMP message.

[6] F. Baker, Ed, 1995, Requirements for IP Version 4 Routers.
[7] Postel, J. (September 1981). Internet Control Message Protocol

The ICMPv4 protocol is initialized using ICMP_init in net / ipv4 / ICMP.py. ICMP cannot be compiled as a module, so there is no module_init or module_cleanup function. The meaning of _ _init can be found by teaching the ICMP_init macro.

2.6. ICMP Messages function

The two categories of ICMP messages presented in the "ICMP Header" section, Errors and Queries, are sent using two different routines:

- icmp_send

The kernel uses it to transmit ICMP error messages when certain conditions are detected.

- icmp_reply

The ICMP protocol uses them to respond to input ICMP request messages that require a response. Both routines receive a buffer in the entry. However, those used as inputs to icmp_send represent the ingress IP packet that resulted in the ICMP message being sent, whereas those in the input to icmp_reply represent the ICMP request message requiring a response.

The code in net/core / icmp.py handles incoming ICMP messages, and therefore it always uses icmp_reply to transmit the ICMP message in response to another message received in the entry. Other kernel subsystems (e.g., routing, IP, etc.) use icmp_send when you need to generate ICMP messages.

2.7. ICMP send and Receiving function

The icmp_rcv is a function called ip_local_deliver_finish to handle the input of ICMP messages. ICMP registers the routine icmp_rcv of the receiver in the net / ipv4 / protocol. First, the checksum of the ICMP message is checked. Note that even when the recipient's NIC can calculate the L4 checksum in hardware (which would be the ICMP checksum) and the checksum says that the ICMP

message is corrupted, icmp_rcv checks the checksum again in the program. Refer to the "Sk_buff Structure."

Not all types of ICMP messages can be sent to a multicast IP address: only ICMP_ECHO, ICMP_TIMESTAMP, ICMP_ADDRESS, and IMCP_ADDRESSREPLY. The icmp_rcv filters those messages that do not respect this rule. In particular, ICMP_ECHO input messages for broadcast are dropped if the system is configured to do so. See the section "Adjusting over the file system / proc."

When all sanity checks are met, icmp_rcv passes the ICMP entry message to the right helper routine. The latter is accessed via the icmp_pointers bus that is initialized at the end of net / ipv4 / icmp.py. The icmp_pointers are a collection of icmp_control data structures. Part of the icmp_pointers configuration. See the "icmp_control Structure" section for the exact meaning of processor and error fields. Any types that are not in the table are outdated, unsupported, or assumed to be processed in the kernel space. For all of these types, the wizard is initialized to icmp_discard.

Chapter 3

Related Work

Chapter 3 : Related Work

Several researchers are still trying to prove that various protocol headers are complicated to exploit incorrectly and to build protection and control systems for networks. They are also trying to find new technologies that keep pace with the rapid development and potential capabilities of hackers and saboteurs, which hurt the security, integrity, availability, and credibility of the information. Areas of detection of data concealment systems in images, in addition to hiding data inside audio clips and others, ensure confidential communications through hidden channels in network protocols and other means that contribute to the detection and tracking. Selected research and studies in this field are summarized below.

In 2004 Buchanan and Llamas presented a research paper to analyze data hiding in secret channels across the server and client where the restrictive broker server, verification rules, etc., were searched. Forward proxy servers use a specific design on the client-side, in one way or another, that accepts that the client knows about its association with external systems that will depend on specific standards and controls. Server, the client sees that the utility should be obtained as usual assistance. In this way, no exceptional design in the electronic customer is required. Depending on the type of reverse intermediate server, the returned material can be as if it were the root itself. This paper proposed a new architecture for information that covers and identifies through a smart data masking agent inserted into a backend broker server. This operator is responsible for covering and detecting exercises, just as the Board anticipates counter-convention operations as a data transfer system. In the current security situation, where almost anything can be considered data, it should also use plans that depend on Stefano parts, which consider hidden parts that can operate in a separate mode.

In 2004, Llamas presented research where he designed a proxy server to hide information in the hidden channel using the definition header inside the IP packet when requesting a specific service from the server. The researcher designed hidden channels using the identification field in the IP

packet and the serial number field by the same mechanism. In the TCP header in a Linux system environment of 2004, there was also a problem of hiding information inside the IP packet header in the Identification field segment offset field if network routers did not fragment the packet [8].

In 2005 Church, et al. presented a research paper hiding data in IP fields for identification and data displacement. The researchers attempted to hide certain information through "secured dialects." The present-day PC world means to hide mysterious messages in any computerized visual and audio signals. Concealment works by replacing portions of unhelpful or unused information in regular computer documents with portions of various.

This study presented results on undetectable data. This hidden data can be explicit content, a personal message, or even pictures. A logical word must be centrally isolated from everything (data to images). The strategies used are expected to make it difficult to perceive that anything is inside the honest record. However, the recipient must have access to the cached information without any problem. The most crucial element of the information hiding framework is how correspondence is permitted between two accredited gatherings without an eyewitness who knows that the correspondence occurs.

The study mode also reviewed TCP/IP agreement used on the Internet. The TCP / IP protocol was created by the Ministry of Defense (MoD) that undertakes to link some different systems organized by different dealers in the IP systems. It is responsible for transferring part of the information from hub to hub. TCP is responsible for verifying the correct transmission of information from the client to the server. The IP agreement describes the basic unit of transmitting information over the Internet as an integral part. All information is merged into IP packets on the sending computer and reassembled on the receiving computer. Each packet begins with a header with control and frame-

[8] David Llamas (2004), Covert Channel Analysis and Detection using Reverse Proxy Servers, School of Computing, Napier University, EH10 5DT, Scotland, UK

tending data. The header packet is isolated in a 20-byte parcel header of information divided into a few fields. Each field has a specific reason, depending on the type of information in the packet payload [(9)].

In 2008, Bharti and Snigdh presented a study on secret communication in the communication system. The researchers studied hiding data or conveying hidden messages in a specific host bus to upgrade the incentive through imperceptible differences. The paper revolves around the existing technologies used with ipv4 and examines different calculations. The transmission of information across the web goes beyond the different layers of TCP and IP agreements. Each layer has its characteristics, which clarifies the situations that can best be utilized. Data can be covered in the vehicle and system layers using discretionary fields, semantic changes, and meanings of low development agreement information units (parcels). Hiding information about the agreement is a plan to bypass the firewall. TCP / IP contains hidden channels accompanying the control of HTML, XML, HTTP, UDP, TCP, IP, ICMP, and Ethernet (CSMA / CD) in the four layers separately [(10)].

In 2007 Zander and Branch presented a study on the hidden channels inside the IP header, specifically in the TTL field, where secret channels are used in the mysterious transfer of data. In contrast to encryption, which protects data only from unauthorized onlookers, secret channels plan to conceal the correspondence's actual existence. The sheer scale of information and the significant number of different system conventions on the Internet makes it an ideal medium for confidential correspondence. Secret channels present a real security risk as they can be used in many harmful exercises. This paper presents a new channel for incognito browsing within the lifetime IP (TTL) address. The sender controls the TTLs for the resulting packets that send confidential data to the recipient. Since the components are organized along the way, and the TTL change, this channel's limit, and confidentiality depend on the "Featured" TTL diversity. We investigate this diversity by following a different peak hour throttle and proposing a coding scheme, making the remote channel

TTL look 'normal.' We are also examining techniques to get rid of this secret channel and to identify it[11].

Chapter 4

Methodology

Chapter 4 : Methodology

In this chapter, we discuss aspects related to some types of cyber-attacks that exploit the ICMP protocol understudy, hidden channels, or areas that can be exploited to leak data in a way that does not have any physical or technical impact using the ICMP protocol. Also, we look at some of the exploits of other protocols that have been done in earlier studies.

We worked to apply the theoretical aspect of this study to the practical side to answer the study questions, prepare the negation particles, and prove the hypotheses related to the study.

This research employs a practical experimental approach in the scientific research process.

In the second chapter, we presented the theoretical part of our study methodology. This chapter will complete the applied study methodology to generate the data presented for each experiment and then start the analysis phase and draw the study sample results using different technical tools.

In this message, one of the ICMP headers will be used to encrypt text data in an encrypted and confidential manner through a valid and reliable request inside one of the protocol headings. The message will be represented in ASCII mode and injected into one of the protocol headers while skipping the surveillance systems. Also, the message to be sent will have a small size and high transmission efficiency and data preservation for transmission within the header. The reason for the use of the ASCII system is that computers store or process only numbers. In this sense, the ASCII code constitutes the numerical representation of letters, numbers, and some commands.

This environment is designed to work in a non-virtual way consisting of systems and code prepared in the Python language and the Internet and the internal network for data to pass through more than one area.

4.1. Applied Study

Before starting the practical side of the study, we must identify some critical elements.

4.1.1. Cyber-attacks using ICMP protocol

In terms of direct or indirect impact, cyber-attacks are considered a source of risk to network and information system managers within an organization regardless of their clarification. Some have identified cyber-attacks as unethical exploitation of computer systems and networks based on technology and communications. The attacker's software capabilities and weaknesses in filling these gaps in the victim to cause material damage, software or moral, may lead to stop the service, leaked data, hacking systems, and encrypting files to pay ransomware or various forms of cyber-crime. Any such form constitutes a threat to information security and the security protection triangle regarding privacy, availability, and coherence.

It has emerged with this development in the field of cyber-attacks what is known as internal and external attackers. The internal attacker is the person or employee within the network who carries out the exploitation using authorized access issued to him. This is the most dangerous type of attacker as he can view and prioritize essential data and tasks, or may facilitate the task of an external attacker in any way, for example by providing him with the types of operating systems of the data center.

Many studies have proven that internal cyber-attacks have a more significant impact than external attacks since the attacker is from the inside as this complicates the audit section to determine the evidence of the attack or follow-up as the internal attacker knows the mechanism and calibration and methods of audit and preservation places that we know.

In this Figure 4.2.1, we review cases of using the ICMP protocol in cyber-attacks and the impact of these attacks.

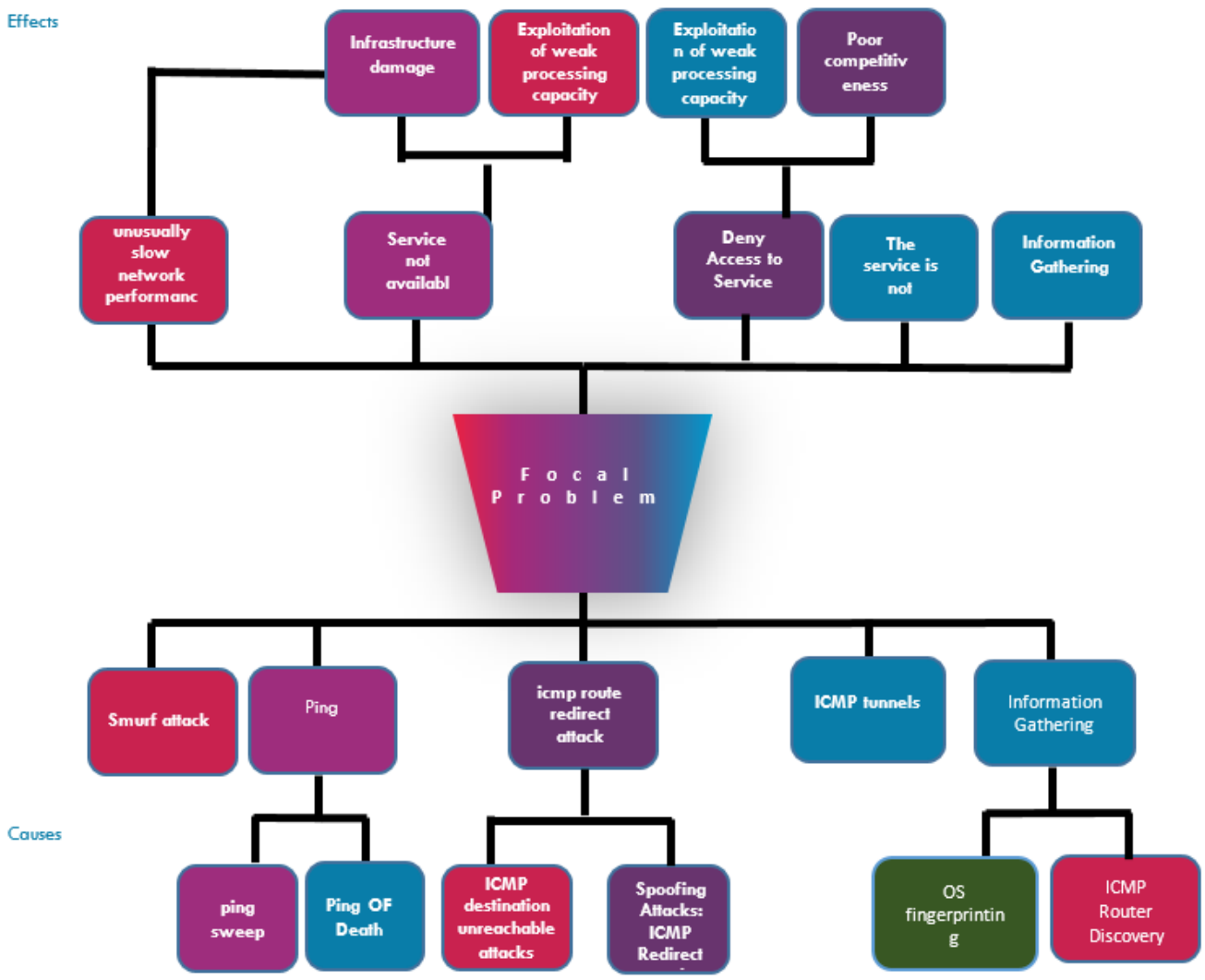


Figure 4.1. ICMP protocol cyber-attacks

For example, in 2019 Google expelled an employee for leaking data and the company's names. The employee deliberately searched for confidential documents and shared them outside the scope of his job.

In the same context of access, Amazon has admitted to dismissing the sales office on the back of investigations into customer data leakage to other marketing companies.

In a related case, 419 million phone numbers associated with Facebook accounts and their Token address were leaked. In the same context of our study of exploiting the ICMP protocol to form cyber-attacks on February 28, 2018, Godthab (a popular developer platform) was attacked by sudden traffic of 1.35 terabytes per second, according to GitHub. Pass to "over a thousand different standalone systems (ASNs) across tens of thousands of unique endpoints." In Figure 4.2, we can see how much difference there is between normal traffic levels and attack levels.

This graph highlights the big difference between normal traffic levels and attack levels [(12)]. GitHub has rapidly amplified its network transport capacity, allowing it to handle some types of related attacks.

In an attack on *Occupy Central* in Hong Kong, there was a DDOS attack in 2014, targeting the democratic voting system. Attackers sent several election servers with many illegal access requests. In a second, this exceeded the capacity to respond to requests, which led to removing some services from work.

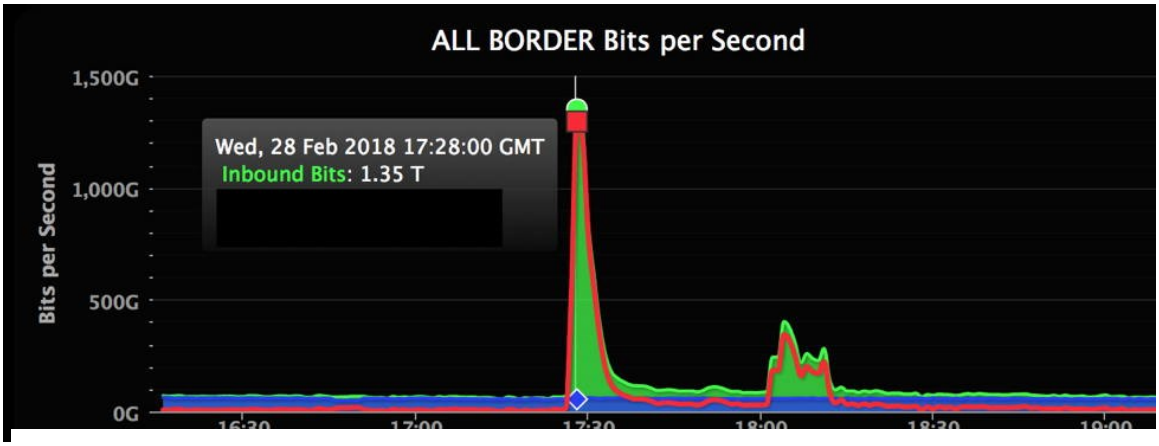


Figure 4.2. GitHub DDoS Attack

The evidence in the troubleshooting process is critical in correcting the problem path to reach the solution and address the problem, not to fall into it. However, the major companies are developing an information security system and electronic risk sensing devices; they have not reached the perfect score due to cyber-attacks proof development.

The exploitation of the protocols in an inappropriate way for its work and not within what is allocated to its work has helped attackers leak data and cut services and piracy, impersonating people and stealing their financial statements and many forms. We will review some of the cyber-attacks based on the ICMP protocol in its composition, including:

4.1.2. ICMP Tunneling

A form of a secret channel created where any security mechanism does not control the flow of information. The ICMP tunnel creates a channel between the client and the server, which forces the firewall not to trigger an alarm if the data is sent via ICMP. An ICMP tunnel is a secret communication between two endpoints using ICMP echo requests and reply packets. So by using an ICMP tunnel, one can pump arbitrary data into the echo packet and send it to a remote computer. The remote computer enters an answer in another ICMP packet and sends it back. This traffic type remains undetectable for a proxy-based firewall, as it focuses more on the source and destination IP address.

These mechanisms can be used to bypass firewall rules by jamming physical traffic. Application-based firewalls can only detect such traffic, as do deep packet inspection on the entire packet. Therefore, network or security administrators will not detect such encrypted connections unless a deep packet inspection is performed.

4.1.3. Fingerprint operating system

A fingerprint is a technique used to determine the operating system the server is running by looking at an ICMP packet response. Now two essential concepts must be remembered by the operating system fingerprint: If the ICMP response has a TTL value of 128, it is a Windows-based device, and if the ICMP response has a TTL value of 64, it is a Linux-based device.

4.1.4. Smurf Attack

This attack is technical as follows: Type 8 is sent, Type 0 is sent again, or an ICMP echo response is sent when an echo request is sent. In a Smurfs attack, the attacker impersonates the ICMP packet source address and sends a broadcast to all computers on that network. If network devices do not filter the traffic, they will be streamed on all network computers. The victim's network is congested by this considerable traffic, which reduces the entire network's productivity.

4.1.5. Countermeasures Smurf Attack

Place filters on routers and firewalls to address counterfeiting addresses. The IP address of the LAN chip must be assigned, and if the IP address of the source device is not in the IP range assigned to the segment, traffic should be dropped. Place filters on L3 devices that do not respond to the broadcast address.

4.1.6. ICMP Router Discovery

One can see that the ICMP router discovery protocol is an IP address for neighboring routers. ICMP router discovery messages are called "router announcements" or "router solicitation." The router discovery message is not a routing protocol. It enables hosts to discover adjacent routers' presence, but a router is not the best way to reach a particular destination. The router declares an ICMP message (type 9, code 0) with a lifetime advertisement. The ICMP router discovery protocol's main challenge is that it does not have any form of authentication. So, end hosts can not know whether the information they receive is valid or not. Because of the above issue, an attacker can lead a man in the middle of an attack wherein the attacker will act as a person in the middle of all connections from the source to the endpoint. ICMP router discovery messages can also impersonate attackers and remotely add bad-route entries to the victim's routing table.

4.1.7. DDoS attack

DDOS is an abbreviation for “Distributed Denial-of-Service Attack,” meaning that a group of computers attacks a single server (one server) to block the service in it. Such attack attempts to make a server or its resources unavailable to the user. This means that when a “DoS attack” occurs, the “Traffic” of the site is consumed, causing the site to stop working and to be unreachable by visitors.

After reviewing some types of cyber-attacks that exploit the ICMP protocol, we will proceed to our study to prove the existence of a new type of cyber-attack that exploits the same protocol.

4.2. What is Covert Channel?

They are areas or paths that can be exploited to not allow the devices inside the networks (the statement is not clear at all), and their contents cannot be analyzed. This is one of the reasons for their poor tracking. Developing systems to deal with this kind of cyber-attack is difficult as covert channels do not have the means to identify suspicious or abnormal behavior.

This study confirmed that covert channels' significant purpose is to transmit information, exploit it in an unethical manner, or use it without being assigned to it, consistent with previous studies.

Covert channels are used for the private transfer of information. Unlike encryption, which only protects unauthorized observers' information, covert channels aim to hide the very existence of the communication. The considerable amount of data and many different network protocols on the Internet make it an ideal high-capacity vehicle for covert communication. Covert channels pose a severe security threat as they can be used for some malicious activities [13]. Channels depend only on the specifications of the protocol to be dealt with since certain conditions and specifications must be available when finding any hidden channels within any header, namely:

1) The packet should appear to be entirely correct according to the internationally agreed size between devices and systems.

2) The packet shall be systematic and consistent with all structural details of its construction.

3) The selection of inappropriate fields in the data packet will record its abnormal passage and facilitate the process of detecting, blocking, or dealing with it in abnormal behavior. Selecting previously discovered fields and developing systems to deal with them will prevent their arrival correctly.

4) If the checksum is correct, then the packet is considered to be reliable for traffic.

4.2.1. Covert channels within some protocols

In previous studies, we have noticed that the research is based on hidden channels and ways of exploiting them in many unique protocols in the network and the protocol under study. Many researchers have also tried to find sophisticated concealment techniques to keep pace with the rapid development of protection techniques and methods of tracking abnormal behavior and concealment in networks' science and exploiting their devices' tools and weaknesses. Some of them performed studies and suggested mechanisms and solutions, including asking questions to link theories to reality later. Some of them put the problem and solution dealing with them for subsequent systems of protection.

This study looked at the importance of the risk of leakage and the effectiveness of systems in tracking this type of data injection within the ICMP packet header. It attempts to answer and many of the raised questions and address the various hypotheses. We will also develop the view of a scientific and technical experiment and build a prototype of the problem and review the solution mechanisms.

4.3. The proposed system for the injection process

Within the ICMP protocol and packet, a data masking system has been proposed within the ICMP header after consideration, research, and many technical experiments on the ICMP packet structure and modification. We use the hidden channels to send the most considerable amount of data regardless of whether the data is encrypted and without modification or disguise.

We have been working on a real and realistic scenario where using a code in Python (accessory **) and an open-source operating system within a fake local network. The experiment was also conducted on a network containing different firewalls in addition to intrusion detection and monitoring systems. We will review some equipment for the working mechanism of the code. There is an explicit working algorithm for the code, as shown in the sender end diagram and a second diagram of the receiver and its working mechanisms. It allows us to transfer data via valid ICMP packets. We will use the client script to pass the data that we want to inject, and then on the device, we are sending to run the server.

Important Notes:

- Anyone monitoring - human or security system - will only see valid ICMP packets.
- There is nothing harmful about the structure of the packets.

In the proposed system, the process of sending any packet requires controls and conditions. It accounts for our experience to succeed and pass the packet reliably and correctly and without any behavior that draws the system's expert attention in protection. It considers the packet size and the number of letters of the transmission, and the number of packets.

Algorithm using hidden channels in network protocols:

1. Input (Network Packets) The data to be sent is hidden.

The algorithm sends process steps:

1. The following libraries and functions are hosted: *argparse*, *sub-process*, *dev null*, and *time sleep.*
2. Calculate the size of the data or file.
3. Question of converting text into ASCII Encoding the text matrix to ASCII system or without change.
4. Each ASCII value will be converted to a binary, and each value of the binary will be sent as a separate packet.

5. Give extra time for the posting task.
6. Determine the future address.
7. Posting Preparation "Dividing data by letter and adding it to our data Array and converting it to a two-way transmission system."
8. Send each letter as a packet or in one packet using the maximum MTU.
9. Open the network port and the transmitter.

Some simple accounts can be used for verification procedures; this will be the formula for calculating and transmitting encrypted data and dividing it into binary data. It will explain how this data is divided into the transmission algorithm. Each ASCII entry (letter, number, and/or symbol) will be divided and converted into binary data. For example, the text encoding process and the time calculation process manually

- This equation tries to reach the expected time without interference or influence from other devices in an approximate manner, with some criteria such as the bandwidth, the connection speed, and so on from the standards

ey a d r e f a i

['1100101', '1111001', '1100001', '1100100', '1110010', '1100101', '1100110', '1100001', '1101001']

N=packet number Start number count =0	N= e y a d r e f a I =8 Packet number = 8*8 =64

The N-value will calculate and transmit unencrypted data, and the packet or data is not divided into the transmission algorithm.

The fact that data is transmitted in its real form ASCII should only take into consideration the criteria:

N=packet number Start number count =0	N= e y a d r e f a I =8 Packet number = 8

2. Outputs: network packets that were injected with data.

The code can be used to capture packets, reformulate, write and convert from ASCII to the original automatically. Alternatively, the Wireshark application can be used to capture the packet and group it in an event there is ASCII encryption and if it is not possible to capture and see the merged text and injected within the packet.

1. We are receiving an algorithm in the case of encrypted text.
2. The algorithm received process steps.
3. The following functions and libraries are called: signal, DateTime, getuid, struct, socket, sys, and streaming socket.
4. Open streaming socket network
5. Capture packets.
6. Finish the capture.
7. Give extra time for the reception task.
8. Begin the process of removing separators and spaces between packets.
9. Data within data Array and aggregated.
10. Convert text from ASCII to simple text.
11. Print text on the screen with reception time and transmitter destination.

4.4. The flowchart of the received process

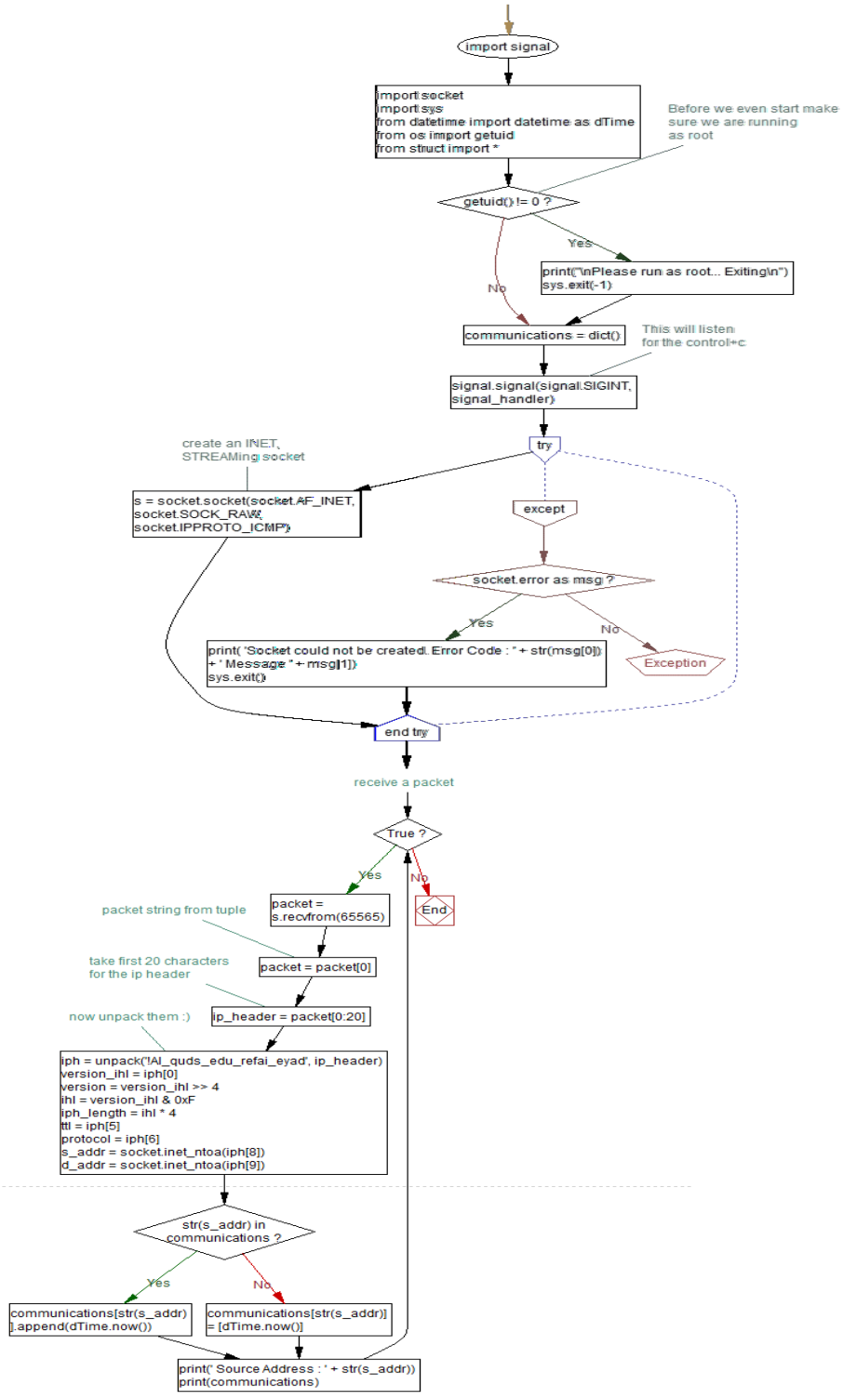


Figure 4.3. The flowcharts of the received process

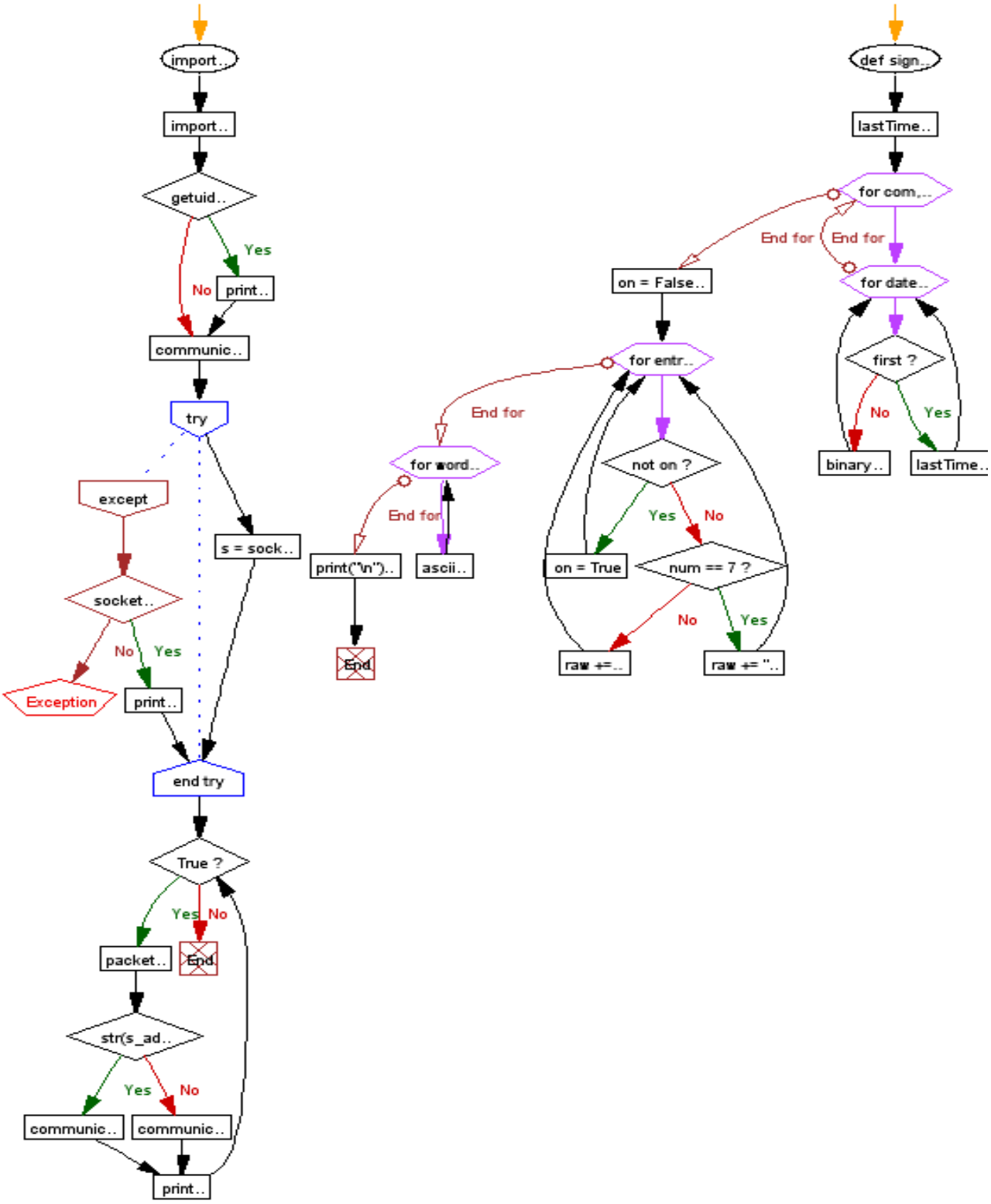


Figure 4.4. Flowchart of the received process

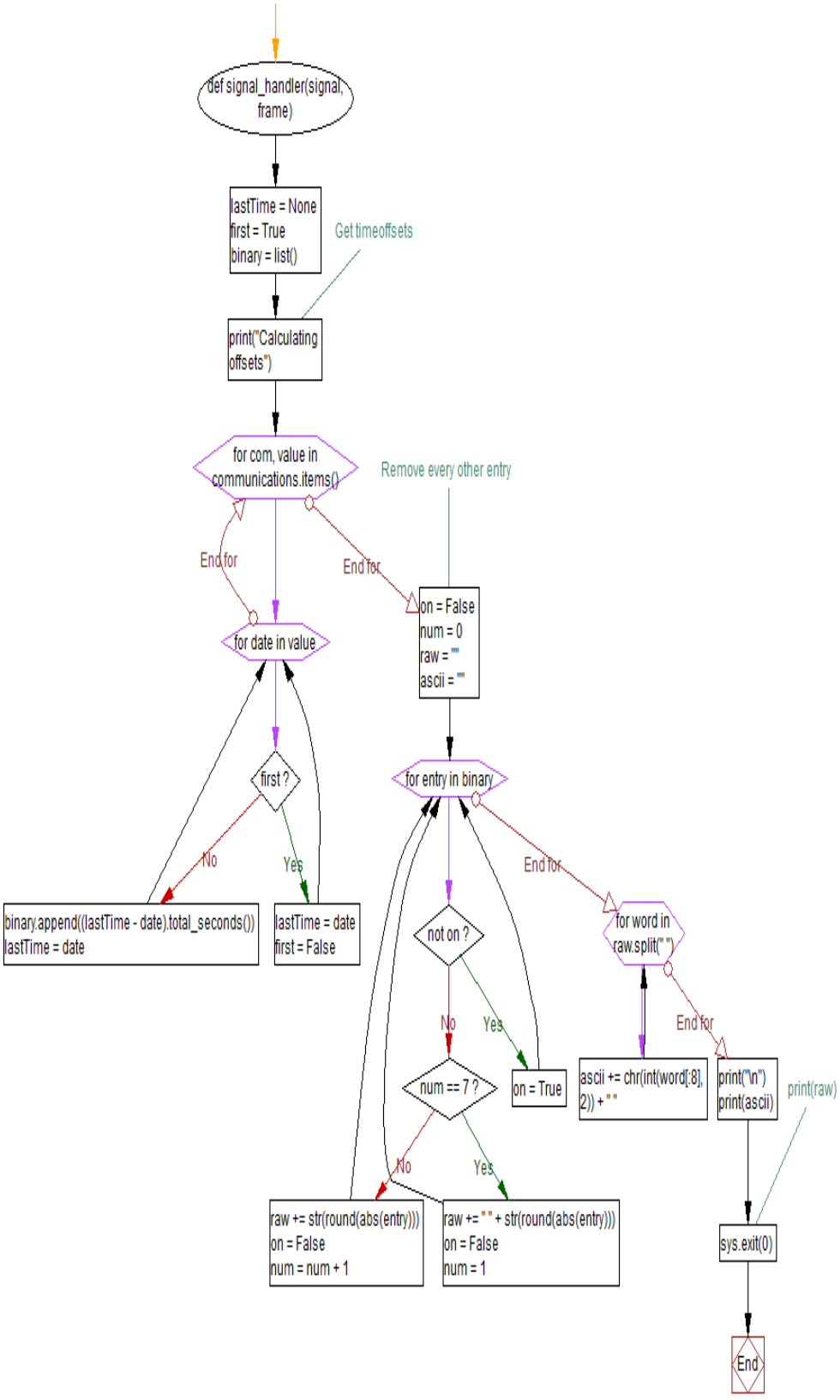


Figure 4.5. Flowchart of the received process

4.5. The flowchart of the sending process

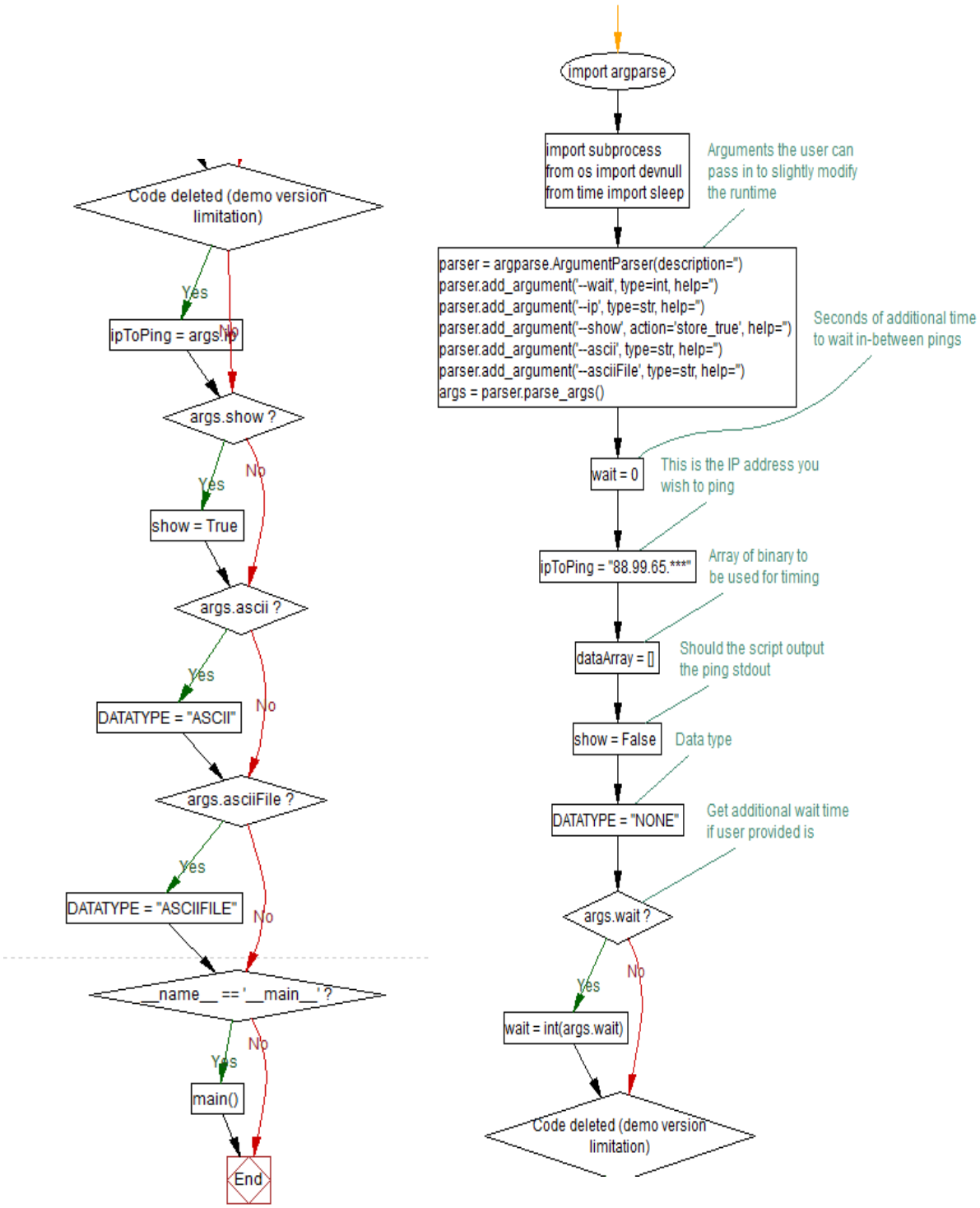


Figure 4.6. Flowchart of the send process

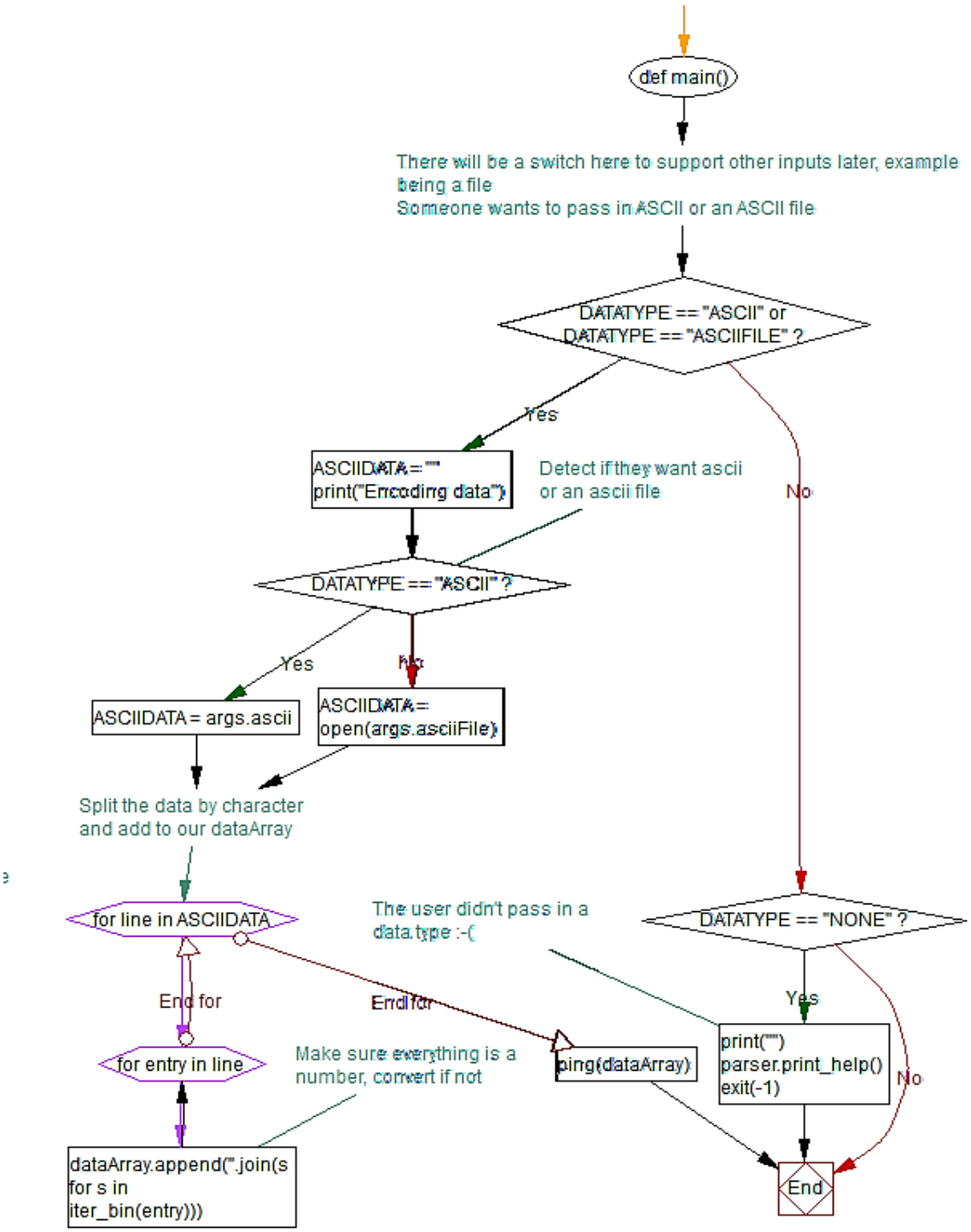


Figure 4.7. Flowchart of the send process

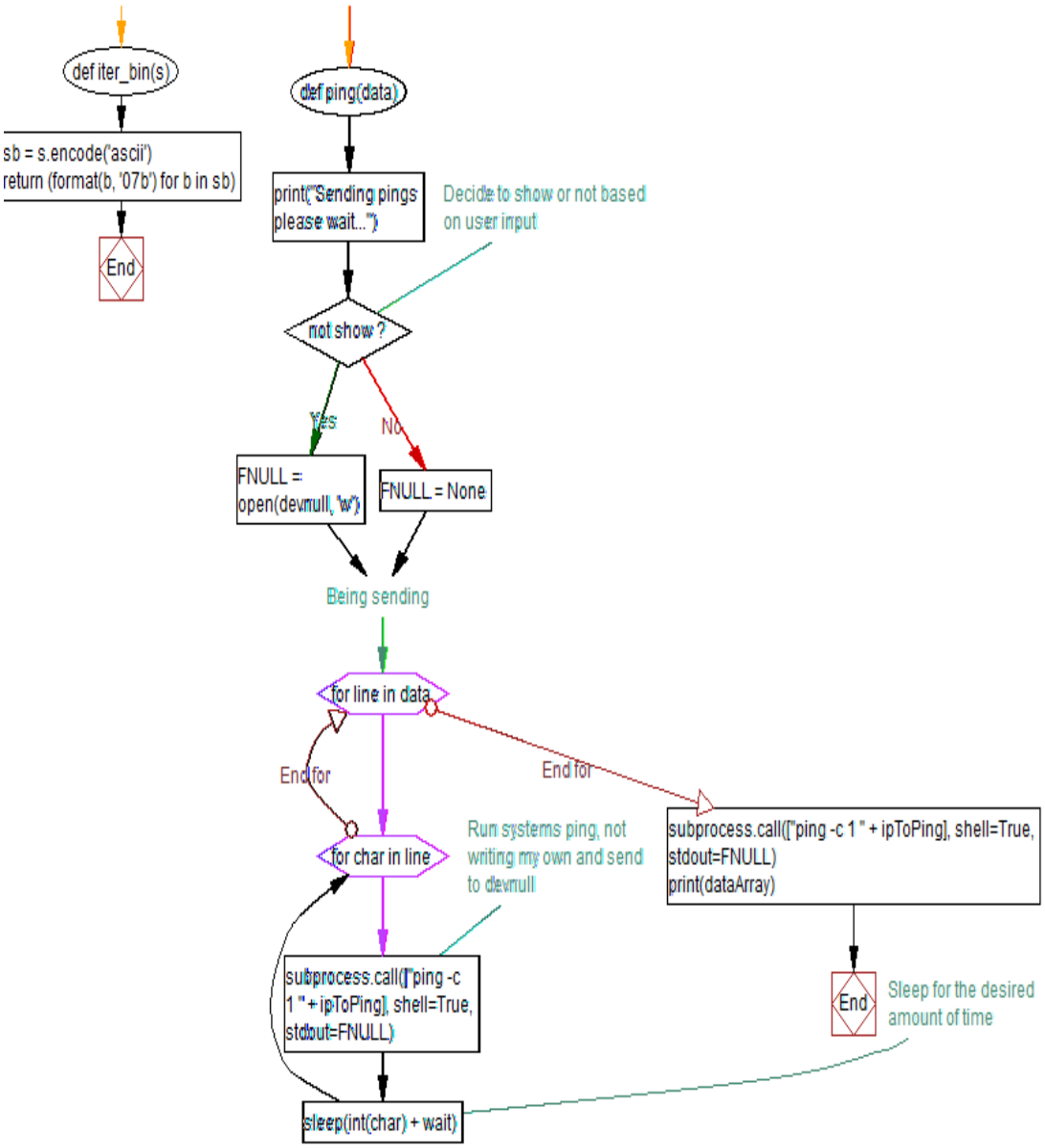


Figure 4.8. Flowchart of the send process

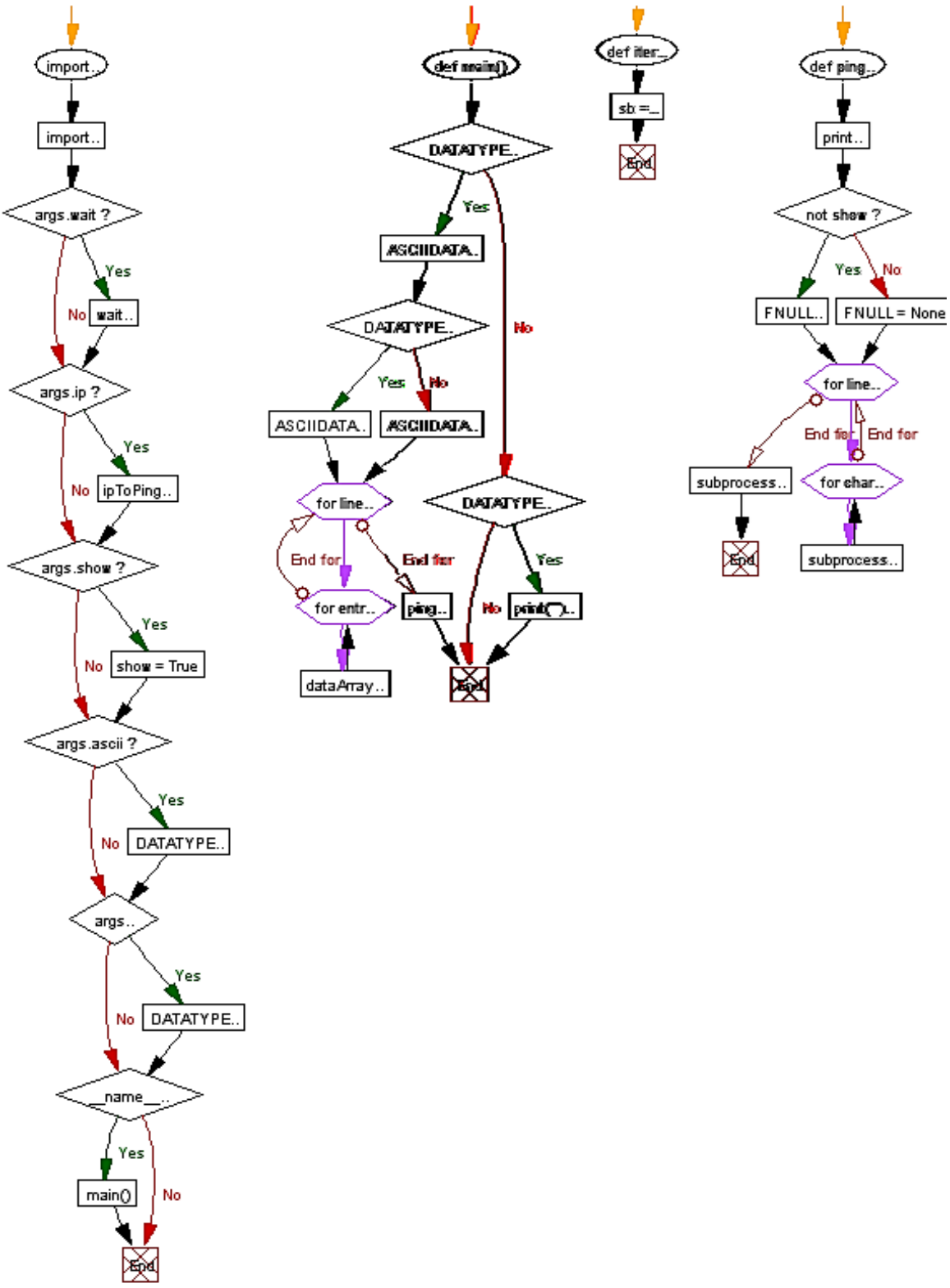


Figure 4.9. Flowchart of the send process

Chapter 5

Simulation Results and Discussion

Chapter 5 : Simulation Results and Discussion

Many protections are taken into account to protect users' data on local networks connected to the Internet from theft, modification, and destruction. The most common protection methods for networks are firewall systems and intrusion detection, which, despite their strength, are vulnerable to several types of attacks. This research focuses on a new strategy and a new type of cyber-attack in concealing, injecting, and abnormal behavior of a packet identified as toxic. What distinguishes this study from other studies include:

- We are adding a new type of cyber-attack using the ICMP protocol.
- We are adding a new type of data leakage through the ICMP network protocols.
- This study uses several fields in the data injection system of the ICMP protocol. It might produce results that could lead to new studies in this field.
- There will be no evidence of data leakage, which is legally essential for the network engineer's maintenance policy.

5.1 Main Simulation Objects

The prototype method will be used in practical work and conducting local and off-site technical experiments using the following tools:

- The Python programming language
- An open-source Operating System (Kali Linux)
- Virtual and real network system (VMware Workstation 15)
- The network analyzer program and capture packets (Wireshark)
- Windows server environment and remote desktop connection

5.2 Experimental Simulation and Results

In this part, technical experiments will be performed and the results will be tabulated. A discussion of the results and analysis will follow.

5.2.1 Experiment No. 1

Experiment No. 1 is a local experiment without a firewall and congestion on the network. We will embed some letters inside the ICMP header to be sent over to the local receiver and then analyze them (Figure 5.1).

```
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 6, 10, 18, 6, 644995), datetime.datetime(2019
^CCalculating offsets


e y a d r e f a i
```

Figure 5.1. Data Sent

The string *"eyadrefai"* was sent as a message encoded through the work environment of the experiment. Figures 5.1 and 5.2 show a snapshot of the data sent and data received in this experiment, respectively.

```
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.043 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.043/0.043/0.043/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.102 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.102/0.102/0.102/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.088 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.088/0.088/0.088/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.513 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.513/0.513/0.513/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.079 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.079/0.079/0.079/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.062 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.062/0.062/0.062/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.063 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.063/0.063/0.063/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.098 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.098/0.098/0.098/0.000 ms
['1100101', '1111001', '1100001', '1100100', '1110010', '1100101', '1100110', '1100001', '1101001']
```

Figure 5.2 Data Received

During the process of sending the packet, operations running on the network were captured. There were three operations of the network working and protocols during the experiment. We note that there was no congestion or overhead processing inside the queue (see Table 5.1).

Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Sent	Packets Received
2019-11-06 15:18:02 Z	C9k713fzCF7dPnSg4	192.168.184.1	60830	255.255.255.255	1947	udp	46.21	80	136	0	0	0	2	0
2019-11-06 15:18:06 Z	CELpnm19v29wVrXuVi	127.0.0.1	8	127.0.0.1	0	icmp	34.95	3584	5376	3584	5376	0	64	64
2019-11-06 15:18:14 Z	C4Mfl43NZvRoQTAPW7	192.168.184.1	60830	192.168.184.255	1947	udp	46.21	80	136	0	0	0	2	0

Table 5.1. Distribution of operations according to services, type of connection and source of connection

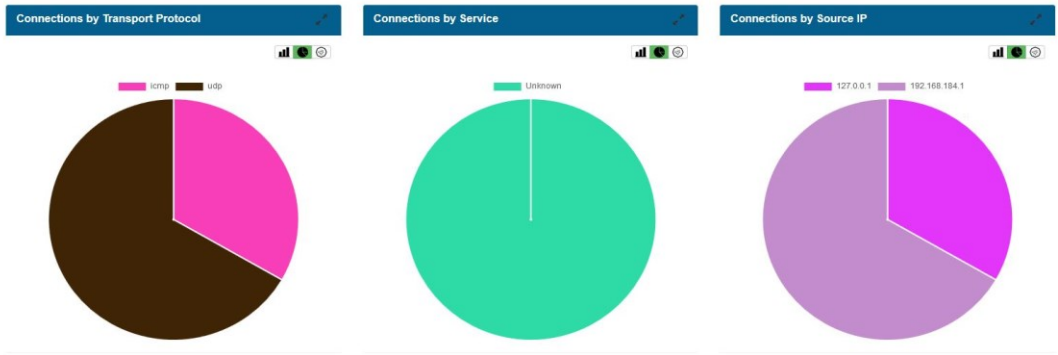


Table 5.2. Distribution of operations according to services, type of connection and source of

Figures 5.3 and 5.4 (I do not see figure 5.4) show the relevant network operations according to their clarification while capturing our packet in the experiment.

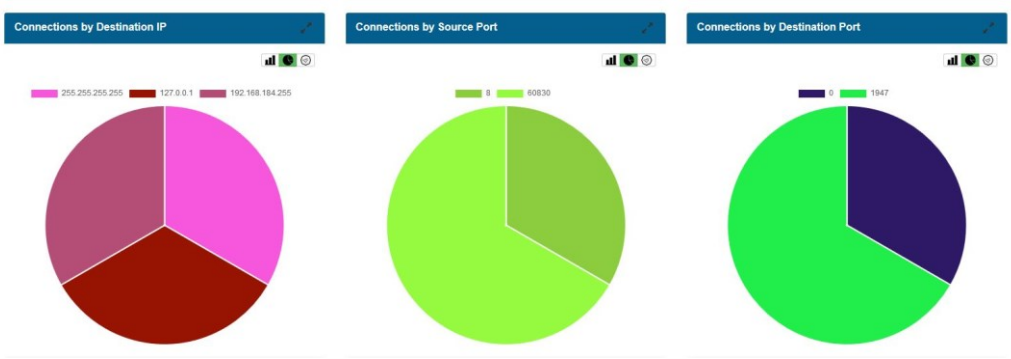


Figure 5.3. Distribution of operations according to services, type of connection and source of connection

In this step, the processes were categorized and filtered, and only the process for ICMP processing was selected and had a connection id (CELpmm19vz9wVrXuVi). Here we filtered and extracted six fields, as shown in Table 5.2.

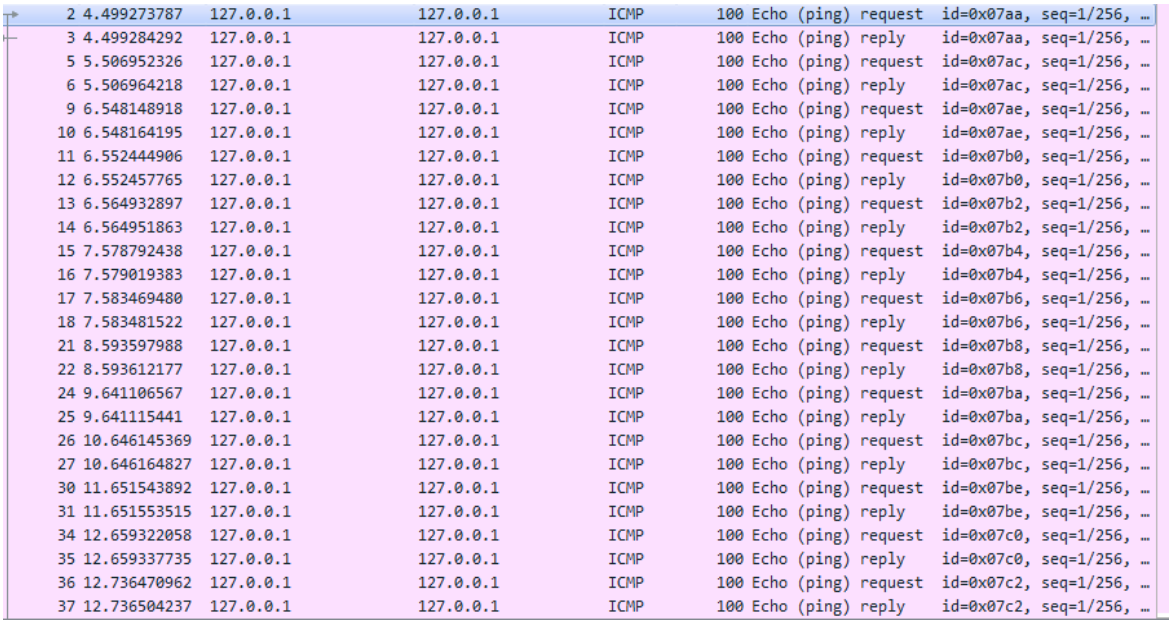
Table 5.2. Experimental data-Test 1

Locally Test Internal Network 3.2													
Meta Data in Payload			e y a d r e f a i		Data Sending			**['1100101', '1111001', '1100001', '1100100', '1110010', '1100101', '1100110', '1100001', '1101001']**					
Tim estamp	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Sent	Packets Received
2019-11-06 15:18:06 Z	127.0.0.1	8	127.0.0.1	0	ICMP	34.95	3584	5376	3584	5376	0	64	64

As shown in the experimental results, we adopt the following variables: the researcher sent each 8-bit number from ASCII to Binary in the form of a separate packet to ensure the ability to analyze and draw conclusions. We notice that 64 packets were sent, i.e. 8 * 8 = 64.

To make sure the system works, we sent an message to the reiver to check that the packet had arrived at the local internet receiver with the same number of transmitted bytes. The process time duration was 34.95 ms. The data size of 3584 payload bytes was received, including the size of the original header. When subtracted from the total bytes sent for all packets (5379), the size of the information for all packets sent is 5379 -3584 = 1792 (total bytes sent - payload bytes sent = original header, i.e., the size of the original header). Thus, the size of the information is 1792 / 64 = 28 bytes (original header/packets received = total packet size). The calculations above were carried out in order to ensure that the exploitation system was working correctly. In the sense that the message was sent in its entirety, and confirmation of the number of bites was sent and received.

As shown in Figure 5.5, we used Wireshark to capture the packet, and we can note that the destination of the packet on my computer sends 127.0.0.1 to the local host. We then sent 32 ICMP request packets to the receiver, and we responded with 32 ICMP reply packets. Therefore, a total of 64 packets were sent and received.



2	4.499273787	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07aa, seq=1/256, …
3	4.499284292	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07aa, seq=1/256, …
5	5.506952326	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07ac, seq=1/256, …
6	5.506964218	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07ac, seq=1/256, …
9	6.548148918	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07ae, seq=1/256, …
10	6.548164195	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07ae, seq=1/256, …
11	6.552444906	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07b0, seq=1/256, …
12	6.552457765	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07b0, seq=1/256, …
13	6.564932897	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07b2, seq=1/256, …
14	6.564951863	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07b2, seq=1/256, …
15	7.578792438	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07b4, seq=1/256, …
16	7.579019383	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07b4, seq=1/256, …
17	7.583469480	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07b6, seq=1/256, …
18	7.583481522	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07b6, seq=1/256, …
21	8.593597988	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07b8, seq=1/256, …
22	8.593612177	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07b8, seq=1/256, …
24	9.641106567	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07ba, seq=1/256, …
25	9.641115441	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07ba, seq=1/256, …
26	10.646145369	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07bc, seq=1/256, …
27	10.646164827	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07bc, seq=1/256, …
30	11.651543892	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07be, seq=1/256, …
31	11.651553515	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07be, seq=1/256, …
34	12.659322058	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07c0, seq=1/256, …
35	12.659337735	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07c0, seq=1/256, …
36	12.736470962	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x07c2, seq=1/256, …
37	12.736504237	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x07c2, seq=1/256, …

Figure 5.4. Picture of the Wireshark network monitoring software system

In Figures 5.6 and 5.7, we verify that the packet arrived correctly.

50 15.813313379 127.0.0.1 127.0.0.1 ICMP 100 Echo (ping) request id=0x07d0, seq=1/256, ttl=64 (reply in 51

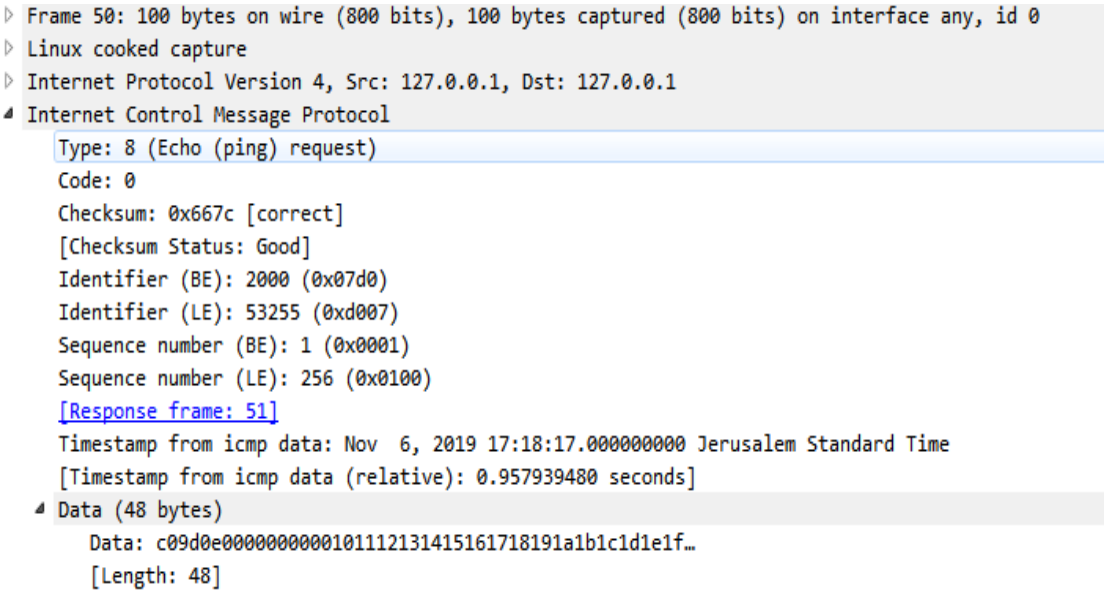


Figure 5.5. Wireshark " checksum packet status" request

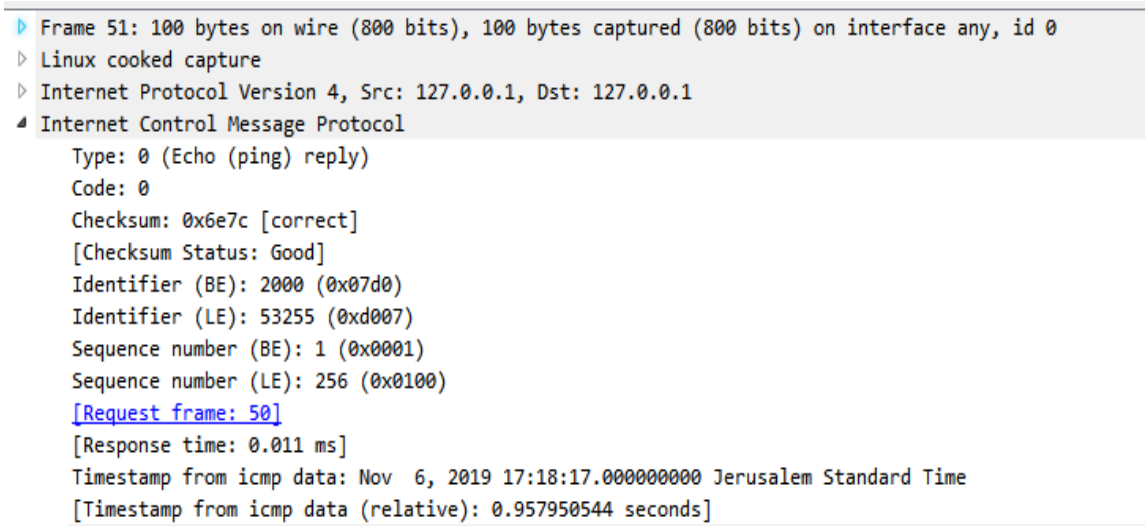


Figure 5.6. Wireshark " checksum packet status" reply

51 15.813324443 127.0.0.1 127.0.0.1 ICMP 100 Echo (ping) reply id=0x07d0, seq=1/256, ttl=64 (request in 50)

Experiment NO.1 Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experimental steps and doing the necessary analysis of the packet, and verifying its correctness, we can conclude that the transmission was 100% successful without any obstacles or data loss. Table 5.3 summarizes these findings.

Firewall	**NO**
data is encrypted	**Yes**
Data before encryption	**eyadrefai**
Data after encryption	**['1100101', '1111001', '1100001', '1100100', '1110010', '1100101', '1100110', '1100001', '1101001']**
Number of operations in the network	**4**
Number of packets for the ICMP process	**64**
RTT	**34.95 ms**
network congestion>50	**NO**

Table 5.3. Results of experiment No. 1

5.2.2 Experiment No. 2

Experiment No. 2 is a local experiment without a firewall and congestion on the network. Therefore, the letters will be sent to the local receiver, and the results will be studied in this part. The message (number 11) was sent encoded through the work environment of the experiment. Figures 5.8 and 5.9 show snapshots of the injected data sent and received in this experiment.

```
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(:
^CCalculating offsets


1 1
```

Figure 5.7. Data Sent

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.193/0.193/0.193/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.044 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.044/0.044/0.044/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.062 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.062/0.062/0.062/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.067 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.067/0.067/0.067/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.024 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.024/0.024/0.024/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.047 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.047/0.047/0.047/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.073 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.073/0.073/0.073/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.065 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.065/0.065/0.065/0.000 ms
['0110001', '0110001']
root@d00m:~/Desktop/ICMP# ^C
```

Figure 5.8. Data Received

During the process of sending the packet, operations running on the network were captured. There were six operations of the network, and we note that there was no congestion or overhead processing inside the queues. (Table 5.4)

	Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Service	Duration	Payload B	Total Byte	Payload B	Total Byte	Missed By	Packets Si	Packets Re	Originate	Tunnel Pa	History
2	2019-11-16 02:11:47 Z	CoxyDr1PNWVVAlzJ4	192.168.184.1	51845	192.168.184.255	1947	udp	null	139.24	160	272	0	0	0	4	0	null	(empty)	D
3	2019-11-16 02:11:48 Z	CSXP5h3cW5i261mI0g	127.0.0.1	8	127.0.0.1	0	icmp	null	6.21	840	1260	840	1260	0	15	15	null	(empty)	null
4	2019-11-16 02:12:25 Z	CJbUhJoLPVrGeFk26	192.168.1.123	68	255.255.255.255	67	udp	dhcp	3	600	656	0	0	0	2	0	null	(empty)	D
5	2019-11-16 02:12:25 Z	CUXBz14VJmCrViLXw3	192.168.184.1	137	192.168.184.255	137	udp	dns	3.65	300	468	0	0	0	6	0	null	(empty)	D
6	2019-11-16 02:12:27 Z	CxKI6j3j5aTm60Y6ii	192.168.1.123	137	192.168.1.255	137	udp	dns	14.48	300	468	0	0	0	6	0	null	(empty)	D
7	2019-11-16 02:12:30 Z	CMM5ok1EmNTACeUv5	192.168.1.123	51845	192.168.1.255	1947	udp	null	92.54	120	204	0	0	0	3	0	null	(empty)	D

Table 5.4. Process schedule for the moment the packet is captured

The following figure (Figures 5.10 and 5.11) show the processes according to their clarification while capturing our packet in the experiment.

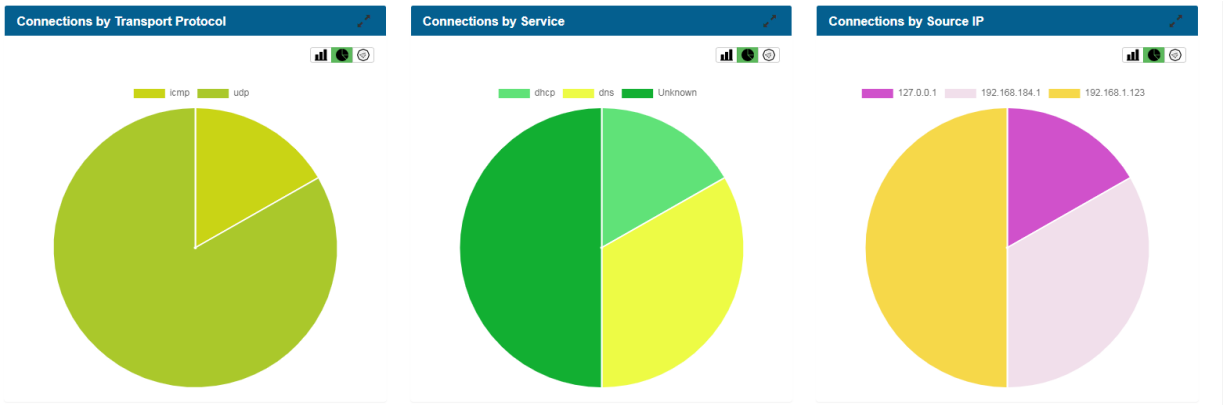


Figure 5.9. Distribution of operations according to services, type of connection and source of connection

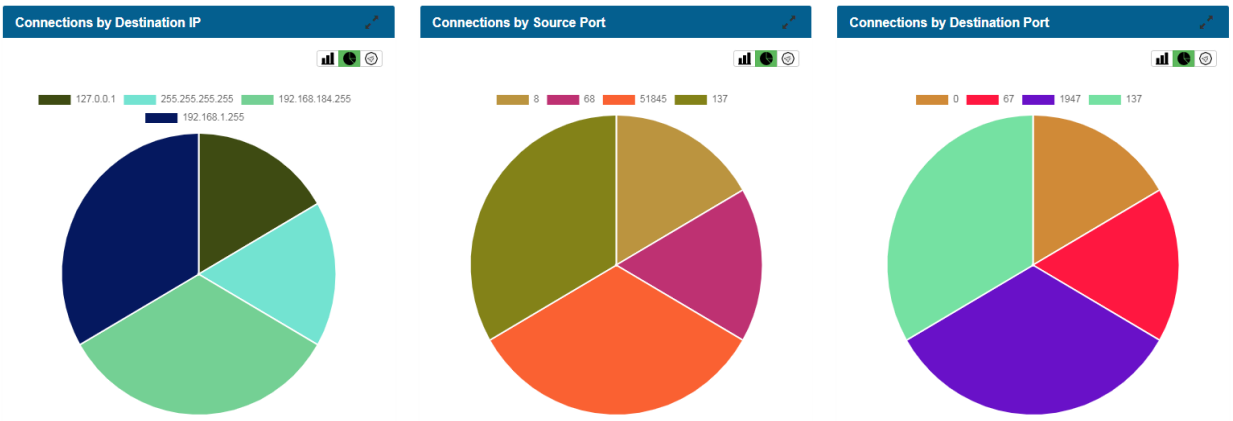


Figure 5.10. Distribution processes, service ports, and IP destination

The processes were categorized and filtered in this step, and only the ICMP process was selected and has a connection id (C5XP5h3cWSIZ61mI0g). In this step, we filtered and extracted six fields, which are shown in Table 5.5.

Table 5.5. Experiment data-Test 2

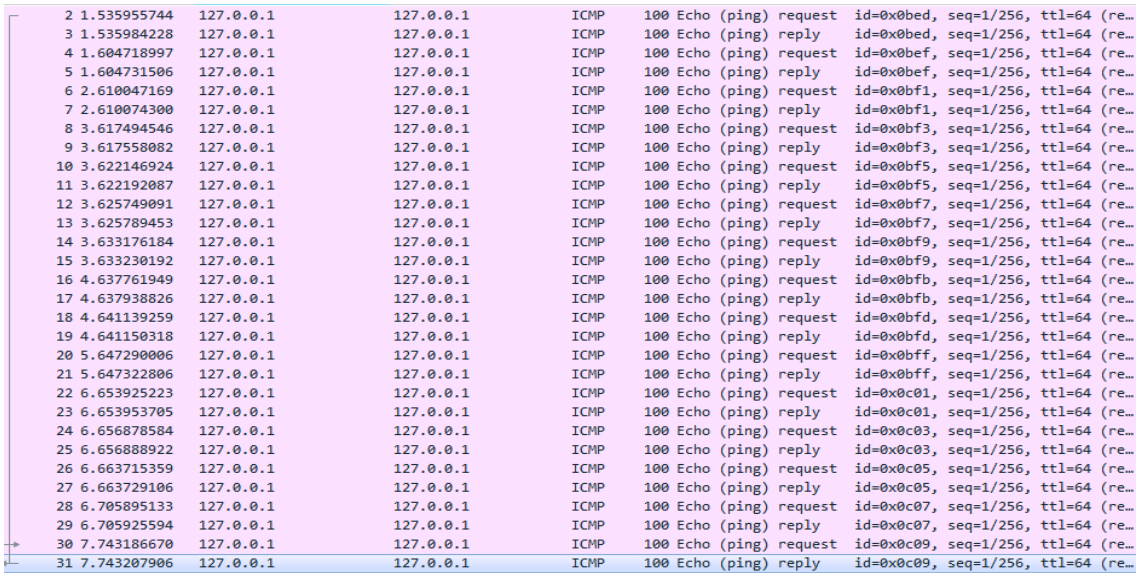
Locally Test Internal Network												
Meta Data in Payload			1 1		Data Sending			['0110001', '0110001']				
Timestamp	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Sent	Packets Received
2019-11-16 02:11: 48 Z	127.0.0.1	8	127.0.0.1	0	ICMP	6.21	840	1260	840	1260	0	15	15

As shown in the experimental results (Table 5.6), we adopt the following variables; the researcher sent each 8-bit number from ASCII to Binary in the form of a separate packet to ensure the ability to analyze and draw conclusions. We notice that 15 packets were sent, i.e. 8 * 2 = 16. To make sure that the system works, we sent a message to the reiver to check that the packet had arrived at the local internet receiver with the same number of transmitted bytes. The process time duration was 6.21ms.

The data size (of 840) payload bytes were received, in addition to the size of the original header. When subtracted from the total size of all packets sent, the original header's size is 1260 - 840 = 420

(Total Bytes Sent- Payload Bytes Sent = original header). The information size is: 420 /15 = 28 bytes (*original header / Packets Received = total size Packets bytes).*

In this step, we are reviewing Wireshark while capturing the packet for the experiment as shown in Figure 5.12. There are certain things to note; firstly, the destination of the packets my computer sends 127.0.0.1 to the local host, secondly we send (16) ICMP requests to the receiver, and we have a response with 16 ICMP reply packets (i.e, a total of (32) sent and received packets).



2	1.535955744	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bed, seq=1/256, ttl=64 (re…
3	1.535984228	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bed, seq=1/256, ttl=64 (re…
4	1.604718997	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bef, seq=1/256, ttl=64 (re…
5	1.604731506	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bef, seq=1/256, ttl=64 (re…
6	2.610047169	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bf1, seq=1/256, ttl=64 (re…
7	2.610074300	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bf1, seq=1/256, ttl=64 (re…
8	3.617494546	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bf3, seq=1/256, ttl=64 (re…
9	3.617558082	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bf3, seq=1/256, ttl=64 (re…
10	3.622146924	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bf5, seq=1/256, ttl=64 (re…
11	3.622192087	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bf5, seq=1/256, ttl=64 (re…
12	3.625749091	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bf7, seq=1/256, ttl=64 (re…
13	3.625789453	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bf7, seq=1/256, ttl=64 (re…
14	3.633176184	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bf9, seq=1/256, ttl=64 (re…
15	3.633230192	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bf9, seq=1/256, ttl=64 (re…
16	4.637761949	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bfb, seq=1/256, ttl=64 (re…
17	4.637938826	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bfb, seq=1/256, ttl=64 (re…
18	4.641139259	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bfd, seq=1/256, ttl=64 (re…
19	4.641150318	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bfd, seq=1/256, ttl=64 (re…
20	5.647290006	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0bff, seq=1/256, ttl=64 (re…
21	5.647322806	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0bff, seq=1/256, ttl=64 (re…
22	6.653925223	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c01, seq=1/256, ttl=64 (re…
23	6.653953705	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c01, seq=1/256, ttl=64 (re…
24	6.656878584	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c03, seq=1/256, ttl=64 (re…
25	6.656888922	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c03, seq=1/256, ttl=64 (re…
26	6.663715359	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c05, seq=1/256, ttl=64 (re…
27	6.663729106	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c05, seq=1/256, ttl=64 (re…
28	6.705895133	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c07, seq=1/256, ttl=64 (re…
29	6.705925594	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c07, seq=1/256, ttl=64 (re…
30	7.743186670	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c09, seq=1/256, ttl=64 (re…
31	7.743207906	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c09, seq=1/256, ttl=64 (re…

Figure 5.11. Picture of the Wireshark network monitoring software system

In Figures 5.13 and 5.14, we verify that the packet arrived correctly.

2, 1.535955744, 127.0.0.1, 127.0.0.1, ICMP 100, Echo (ping) request id=0x0bed, seq=1/256, ttl=64 (reply in 3)

3 1.535984228 127.0.0.1 127.0.0.1 ICMP 100 Echo (ping) reply id=0x0bed, seq=1/256, ttl=64 (request in 2)

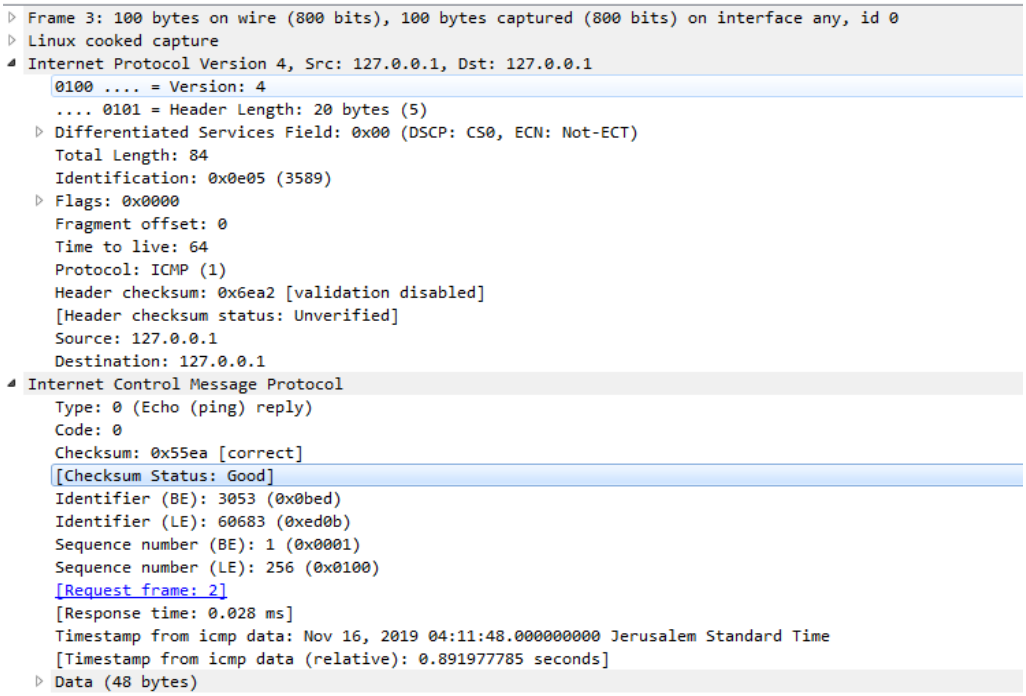


Figure 5.13. Wireshark “checksum packet status" request

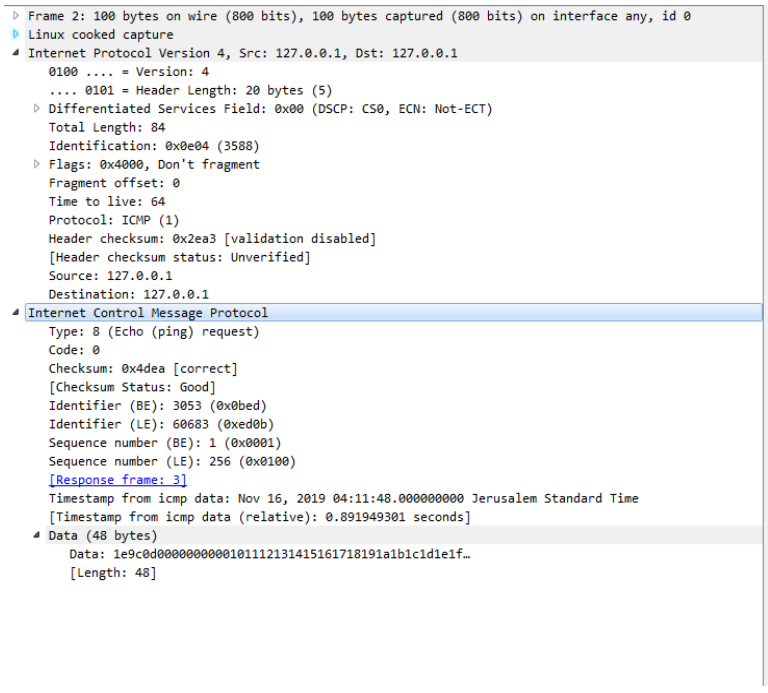


Figure 5.12. Wireshark " checksum packet status” reply

Experiment NO.2 Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experimental steps and doing the necessary analysis of the packet, and verifying the correct access to it, experiment No. 2 did not lose any of the packets. The transmission was 100% without any obstacle or loss. Table 5.7 shows the relevant variables for this experiment.

Firewall	**NO**
data is encrypted	**Yes**
Data before encryption	**11**
Data after encryption	**['0110001', '0110001']**
Number of operations in the network	**6**
Number of packets for the ICMP process	**16**
RTT	**6.21 ms**
network congestion>50	**NO**

Table 5.5. Results of experiment No. 2

5.2.3 Experiment No. 3

Experiment No. 3 is a local experiment **without a firewall or congestion** on the network so that letters will be sent to the local receiver, and the results will be studied. The value (11) was sent encoded through the work environment of the experiment. Figures 5.15 and 5.16 show snapshots of the injected data sent and received in this experiment, respectively.

```
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
 Source Address : 127.0.0.1
{'127.0.0.1': [datetime.datetime(2019, 11, 15, 21, 11, 48, 892120), datetime.datetime(2019, 11,
^CCalculating offsets


1 1
```

Figure 5.14. Data sent

```
60
61  --- localhost ping statistics ---
62  1 packets transmitted, 1 received, 0% packet loss, time 0ms
63  rtt min/avg/max/mdev = 0.107/0.107/0.107/0.000 ms
64  PING localhost (127.0.0.1) 56(84) bytes of data.
65  64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.074 ms
66
67  --- localhost ping statistics ---
68  1 packets transmitted, 1 received, 0% packet loss, time 0ms
69  rtt min/avg/max/mdev = 0.074/0.074/0.074/0.000 ms
70  PING localhost (127.0.0.1) 56(84) bytes of data.
71  64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.058 ms
72
73  --- localhost ping statistics ---
74  1 packets transmitted, 1 received, 0% packet loss, time 0ms
75  rtt min/avg/max/mdev = 0.058/0.058/0.058/0.000 ms
76  PING localhost (127.0.0.1) 56(84) bytes of data.
77  64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.026 ms
78
79  --- localhost ping statistics ---
80  1 packets transmitted, 1 received, 0% packet loss, time 0ms
81  rtt min/avg/max/mdev = 0.026/0.026/0.026/0.000 ms
82  PING localhost (127.0.0.1) 56(84) bytes of data.
83  64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.055 ms
84
85  --- localhost ping statistics ---
86  1 packets transmitted, 1 received, 0% packet loss, time 0ms
87  rtt min/avg/max/mdev = 0.055/0.055/0.055/0.000 ms
88  PING localhost (127.0.0.1) 56(84) bytes of data.
89  64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.023 ms
90
91  --- localhost ping statistics ---
92  1 packets transmitted, 1 received, 0% packet loss, time 0ms
93  rtt min/avg/max/mdev = 0.023/0.023/0.023/0.000 ms
94  ['1000101', '1011001']
95  root@d00m:~/Desktop/ICMP#
```

Figure 5.15. Data Received

During the process of sending the packet, operations running on the network were captured. There were (5) operations of the network protocols. We note that there was no congestion or processing overhead inside the queues. (See Table 5.7).

Table 5.7.Process schedule for the moment the packet is captured

Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes	Total Bytes	Packets Sent	Packets Received
2019-6 1:47 Z	CoxyDr1PNWrVAIiz34	192.168.184.1	51845	192.168.184.255	1947	udp	139.24	160	272	0	0	4	0
2019-6 1:48 Z	C5XP5h3cWSIZ61mI0g	127.0.0.1	8	127.0.0.1	0	icmp	6.21	840	1260	840	1260	15	15
2019-6 2:25 Z	CJbLhJoLPVrGeFk26	192.168.1.123	68	255.255.255.255	67	udp	3	600	656	0	0	2	0
2019-6 2:25 Z	CUXBz14VJmCrViLXw3	192.168.184.1	137	192.168.184.255	137	udp	3.65	300	468	0	0	6	0
2019-6 2:27 Z	CxKIdj3j5aTmG0Y6ii	192.168.1.123	137	192.168.1.255	137	udp	14.48	300	468	0	0	6	0
2019-6 2:30 Z	CMMSok1EmNTACeLIv5	192.168.1.123	51845	192.168.1.255	1947	udp	92.54	120	204	0	0	3	0

Figures 5.17 and 5.18 show the processes according to their clarification while capturing our packets in the experiment.

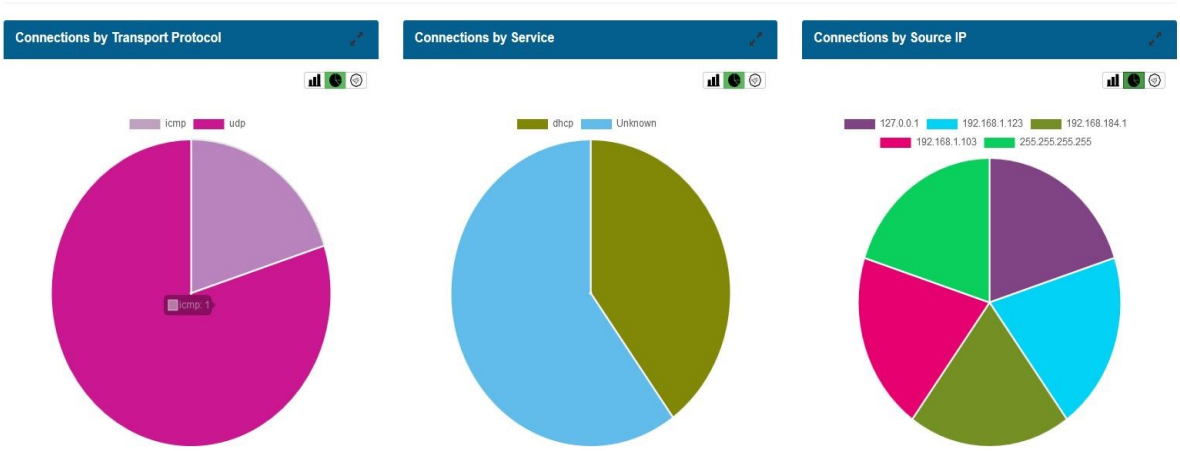


Figure 5.17. Distribution of operations according to services, type of connection and source of connection

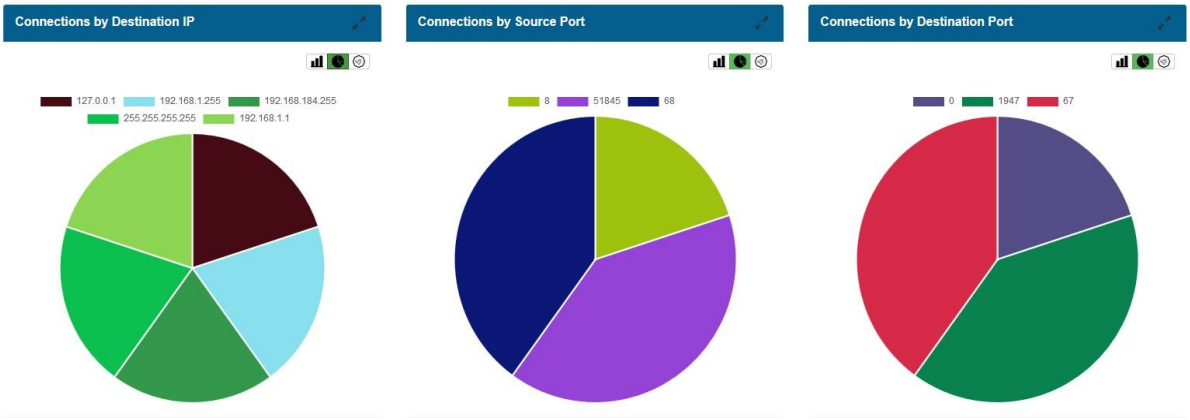


Figure 5.16. Distribution processes, service ports, and IP destination

The processes were categorized and filtered in this step, and only the ICMP process was selected and had a connection id (CELpmm19vz9wVrXuVi). We filtered and extracted (6) fields, as shown in Table 5.8.

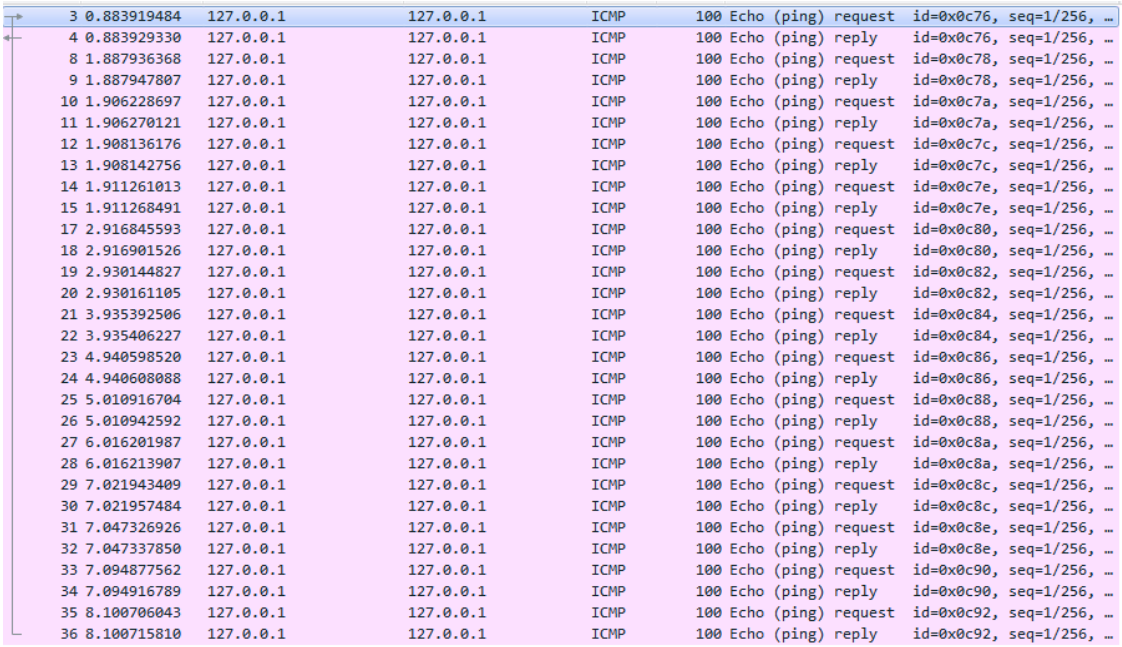
Table 5.8. Experiment data-Test 3

Locally Test Internal Network 3.4													
Meta Data in Payload		E Y			Data Sending			['1000101', '1011001']					
Timestamp	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Sent	Packets Received
2019-11-16 02:14:24 Z	127.0.0.1	8	127.0.0.1	0	ICMP	7.22	840	1260	840	1260	0	15	15

As shown in the experimental results (Table 5.9), we adopt the following variables; in this experiment, the researcher sent each 8-bit number from ASCII to binary in the form of a separate packet to ensure the ability to analyze. We notice that (15) packets were sent, i.e. 8 * 2 = 16. To ensure the system works, we sent a (ping) message to the receiver to check that the packet had already arrived at the local internal receiver with the same number of transmitted bytes. The process time duration is 7.22 seconds.

The data size (of 840) payload bytes were received, in addition to the size of the original header. When subtracted from the total size, the original header's size is (1260 – 840 = 420) *(Total Bytes Sent- Payload Bytes Sent = original header)* for all packets sent. Therefore, the size of the information is (420 / 15 = 28) bytes **(***original header/ Packets Received = total size in bytes).*

We did the analysis using Wireshark, as shown in Figure 5.19, to capture the packets for the experiment. Here are some observations: first, my computer sets IP (127.0.0.1) to the local host, then we send (16) ICMP requests to the receiver, and we have a response with (16) ICMP reply packets that is a total of (32) sent and received packets.



3	0.883919484	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c76, seq=1/256, …
4	0.883929330	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c76, seq=1/256, …
8	1.887936368	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c78, seq=1/256, …
9	1.887947807	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c78, seq=1/256, …
10	1.906228697	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c7a, seq=1/256, …
11	1.906270121	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c7a, seq=1/256, …
12	1.908136176	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c7c, seq=1/256, …
13	1.908142756	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c7c, seq=1/256, …
14	1.911261013	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c7e, seq=1/256, …
15	1.911268491	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c7e, seq=1/256, …
17	2.916845593	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c80, seq=1/256, …
18	2.916901526	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c80, seq=1/256, …
19	2.930144827	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c82, seq=1/256, …
20	2.930161105	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c82, seq=1/256, …
21	3.935392506	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c84, seq=1/256, …
22	3.935406227	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c84, seq=1/256, …
23	4.940598520	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c86, seq=1/256, …
24	4.940608088	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c86, seq=1/256, …
25	5.010916704	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c88, seq=1/256, …
26	5.010942592	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c88, seq=1/256, …
27	6.016201987	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c8a, seq=1/256, …
28	6.016213907	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c8a, seq=1/256, …
29	7.021943409	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c8c, seq=1/256, …
30	7.021957484	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c8c, seq=1/256, …
31	7.047326926	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c8e, seq=1/256, …
32	7.047337850	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c8e, seq=1/256, …
33	7.094877562	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c90, seq=1/256, …
34	7.094916789	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c90, seq=1/256, …
35	8.100706043	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) request id=0x0c92, seq=1/256, …
36	8.100715810	127.0.0.1	127.0.0.1	ICMP	100	Echo (ping) reply id=0x0c92, seq=1/256, …

Figure 5.18. Picture of the Wireshark network monitoring software system

In Figures 5.20 and 5.21, we verify that the packet arrived correctly.

3 0.883919484 127.0.0.1 127.0.0.1 ICMP 100 Echo (ping) request id=0x0c76, seq=1/256, ttl=64 (reply in 4)

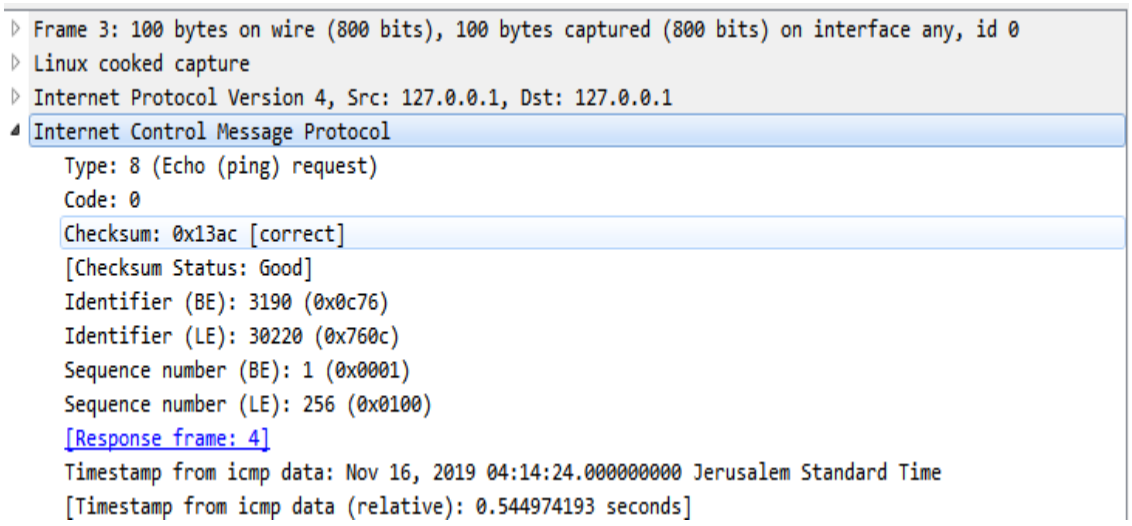


Figure 5.19. Wireshark " checksum packet status" request

4 0.883929330 127.0.0.1 127.0.0.1 ICMP 100 Echo (ping) reply id=0x0c76, seq=1/256, ttl=64 (request in 3)

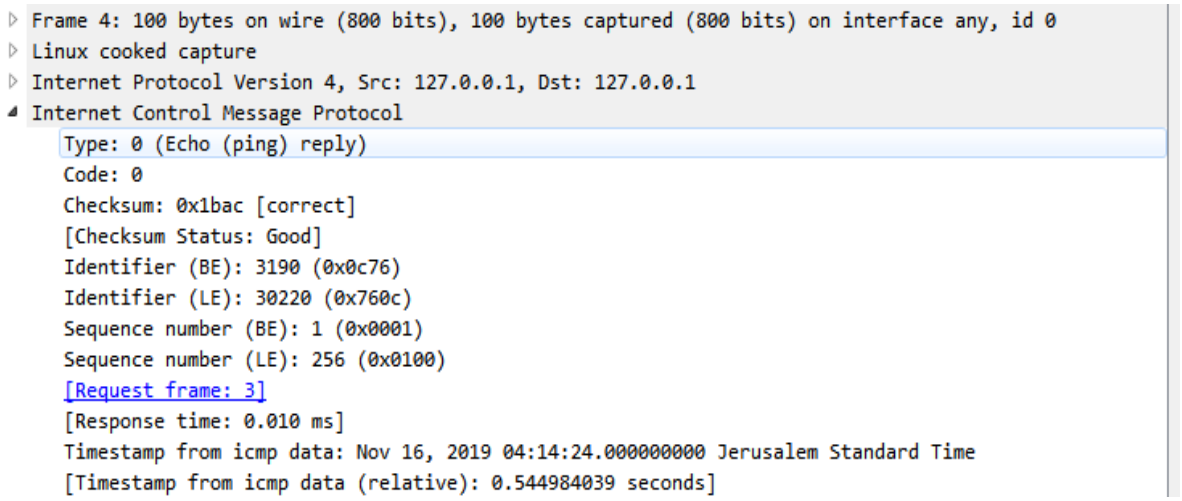


Figure 5.20. Wireshark " checksum packet status" reply

Experiment NO.3 Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experimental steps and doing the necessary analysis and correctness checks, it can be seen that the transmission was 100% correct without any obstacles or loss of data. Table 5.9 shows the relevant variables for this experiment.

Firewall	NO
data is encrypted	Yes
Data before encryption	E Y
Data after encryption	['1000101', '1011001']
Number of operations in the network	6
Number of packets for the ICMP process	16
RTT	7.22 ms
network congestion>50	NO

Table 5.9. Results of experiment No. 3

5.2.4 Experiment No. 4

Experiment No. 4 is an external experiment to a receiver outside the network with a juniper type firewall. We will send a message of seven letters to the external receiver, and the results will be studied in this part.

- The message “CALL_ME” consists of 7 letters sent without “encoding” and without being divided into parts and through the experiment work environment.
- According to the following path, the packet passed through (13) global routers until it reached its final destination, as shown in Figure 5.22 & Table 5.10.

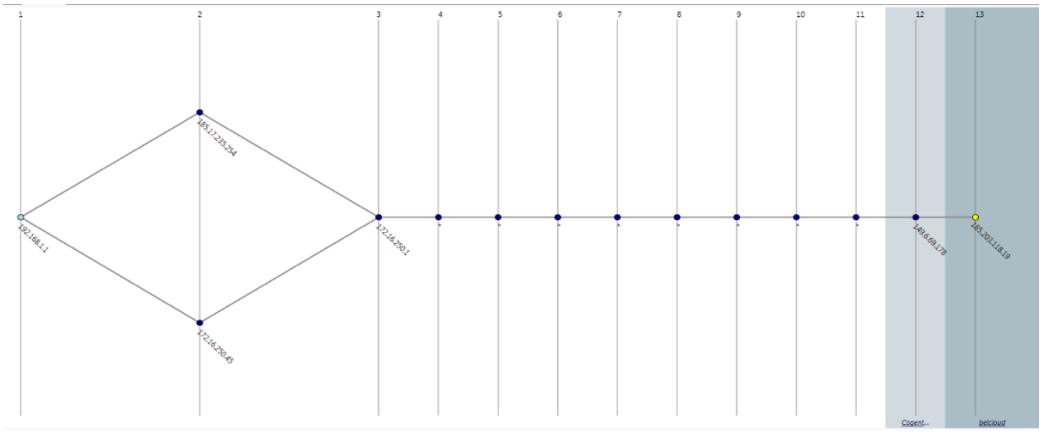


Figure 5.21. Path data flowchart

Hop	IP Address	Hostname	Time (ms)	Country	Status
1	192.168.1.1	?	1	Unassigned or assigned to IANA.org	11:0:The hop limit expired in transit
2	185.17.235.254	ADSL-185.17.235.254.mada.ps	75	PALESTINIAN TERRITORY	11:0:The hop limit expired in transit
3 4 5 6 7 8 9	*	*	*	*	*
10	*	*	*	*	*
11					
12	149.6.69.178	149.6.69.178	82	Cogent Communications	
	185.203.118.19	185.203.118.19	103	belcloud	

Table 5.10. The table shows the data traffic and the addresses of the global entities that these packets pass-through

Map and traffic data packets are divided into three local lines, global Internet zones, and the host data center server for Test 4 (Figure 5.23). Figure 5.24 shows a screenshot of the analysis software and indicates the presence of the firewall inside the network of the juniper receiver server.

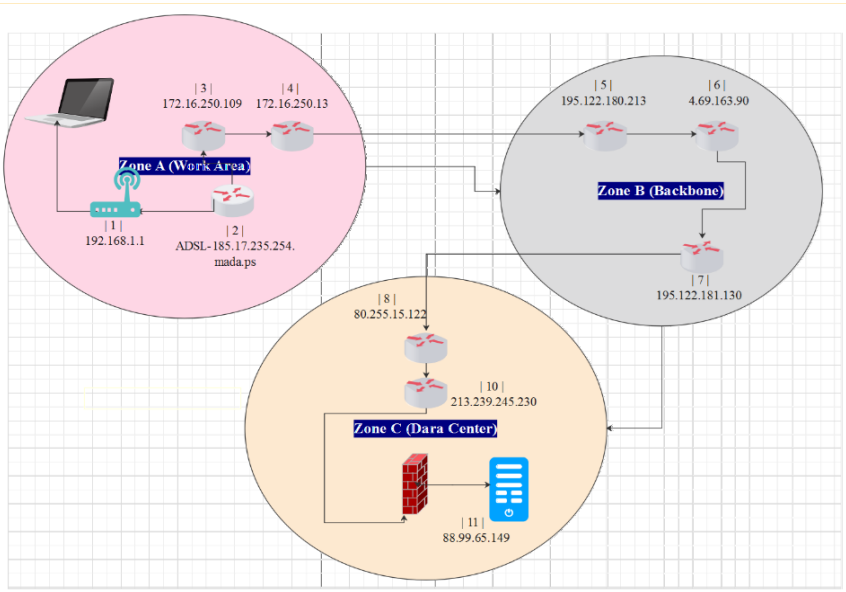


Figure 5.22. Traffic data packets

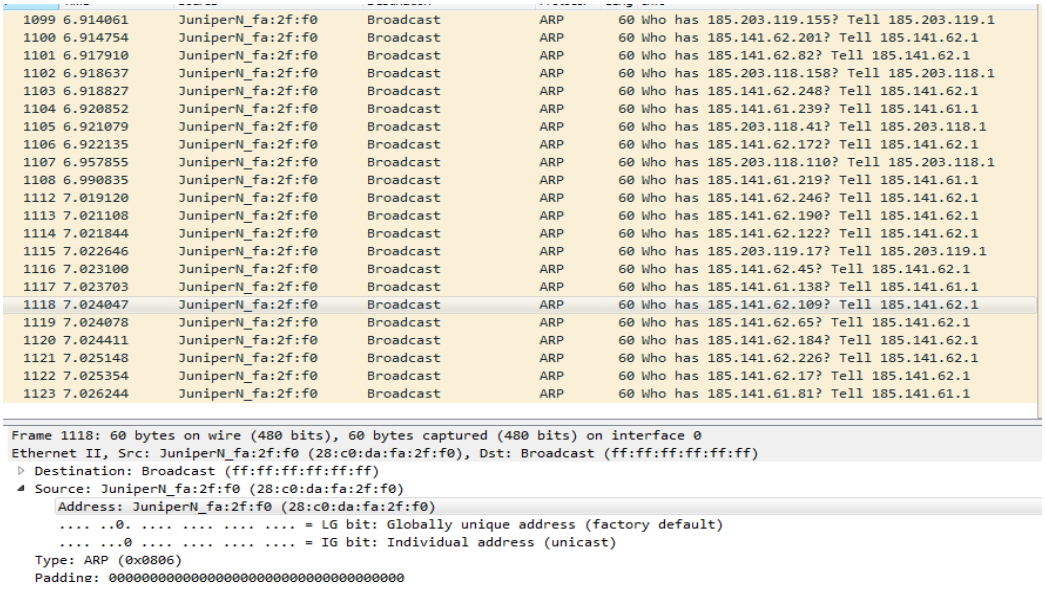


Figure 5.23. A screenshot of Wireshark

During sending the unencrypted packet, operations running on the network were captured inside the recipient's machine, and there were (98) operations. We noted congestion inside the queues and some processing overhead, as shown in the following figures. Figure 5.25 shows the operations according to their clarification while capturing our packet in the experiment “Distribution of operations according to services, type of connection, and source of connection.”

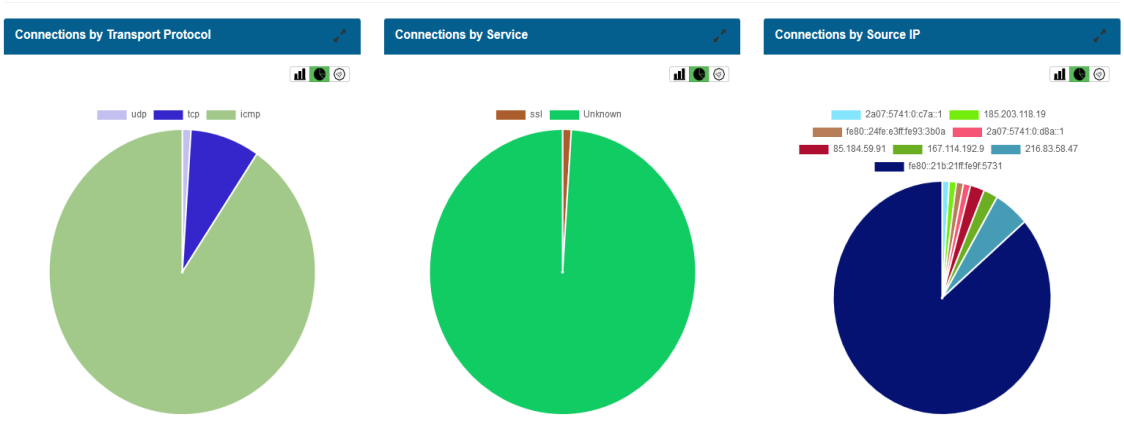


Figure 5.24. Operations during packet capture

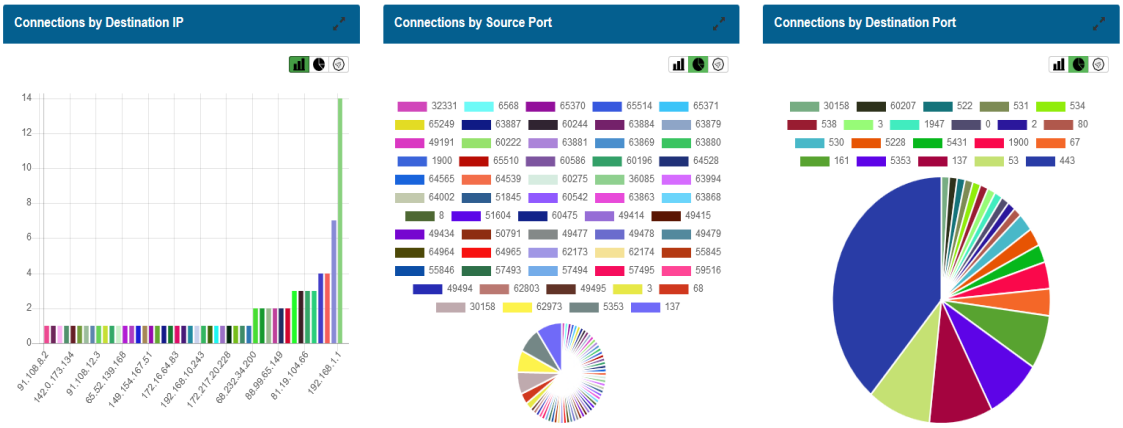


Figure 5.25. Processing operations within the network

In this step, the processes were categorized and filtered, and only the process for ICMP processing was selected and had a connection id (CkZJCk4HJiQRw0NJAk). In this step, we filtered and extracted (6) fields shown in Table 5.11.

Table 5.11. Experiment data-Test 4

Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Transport Protocol	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Sent	Packets Received
11-26 13:33:05 Z	aPKD2HesVThCpp3c	5.184.59.91	8	85.203.118.19	icmp	7	35	7	35	0	1	1

In the experiment, we adopt the following variables: injecting the seven letters message into the ICMP packet, as shown in Figure 5.27. The packet arrived correctly as required without any loss. The time between sending and receiving was very little, 0.525 Ms.

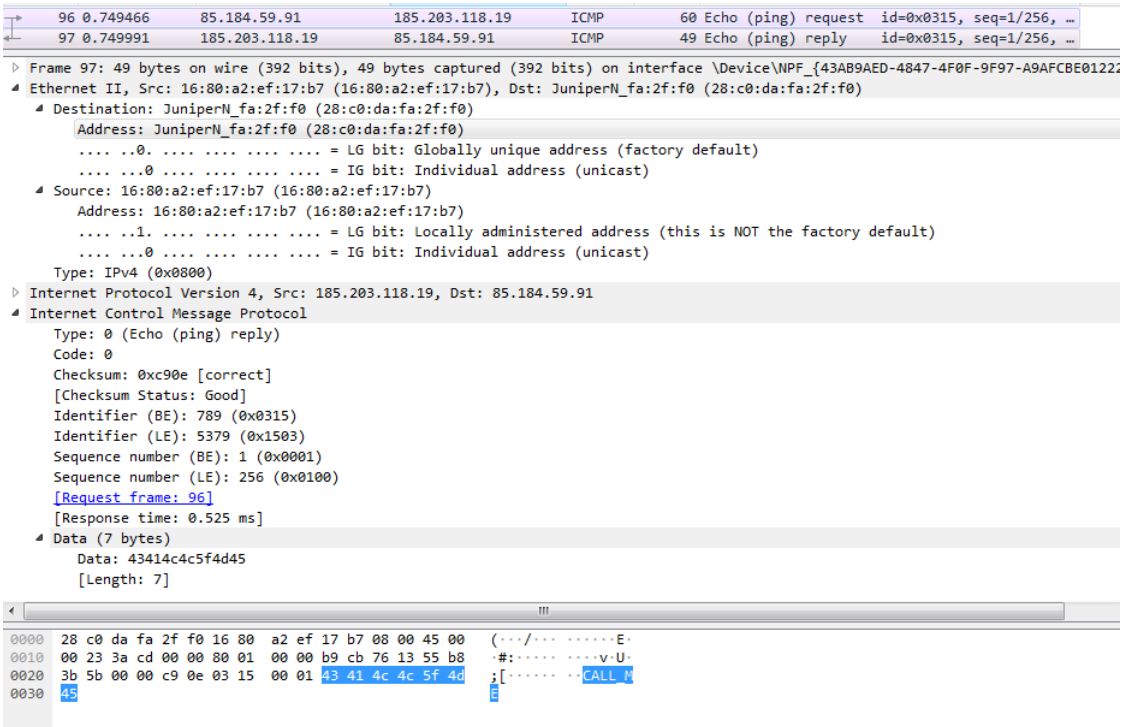
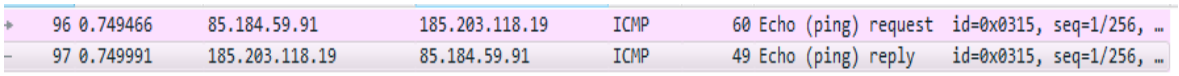


Figure 5.26. The figure shows an image of the packets after transmission and data injection

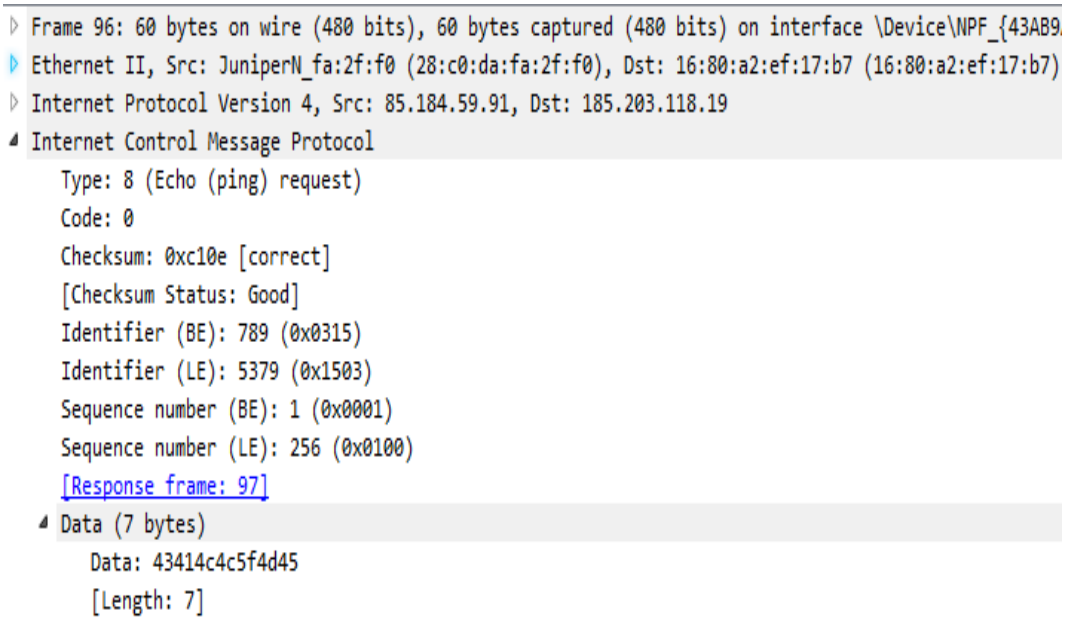
A review of the Wireshark while capturing the packet, we note the following: firstly, the destination of the packets my computer sends (85.184.59.91) to (185.203.118.19). Secondly, we send one ICMP request to the receiver, and we have a response with one ICMP reply; a total of 2 sent and received



packets. These findings are shown in Figures 5.28, 5.29, and 5.30.

Figure 5.27. Picture of the Wireshark network monitoring software system

96 0.749466 85.184.59.91 185.203.118.19 ICMP 60 Echo (ping) request id=0x0315,



seq=1/256, ttl=123 (reply in 97)

Figure 5.28. Figure 5.31 Wireshark " checksum packet status" request

97 0.749991 185.203.118.19 85.184.59.91 ICMP 49 Echo (ping) reply id=0x0315,

seq=1/256, ttl=128 (request in 96)

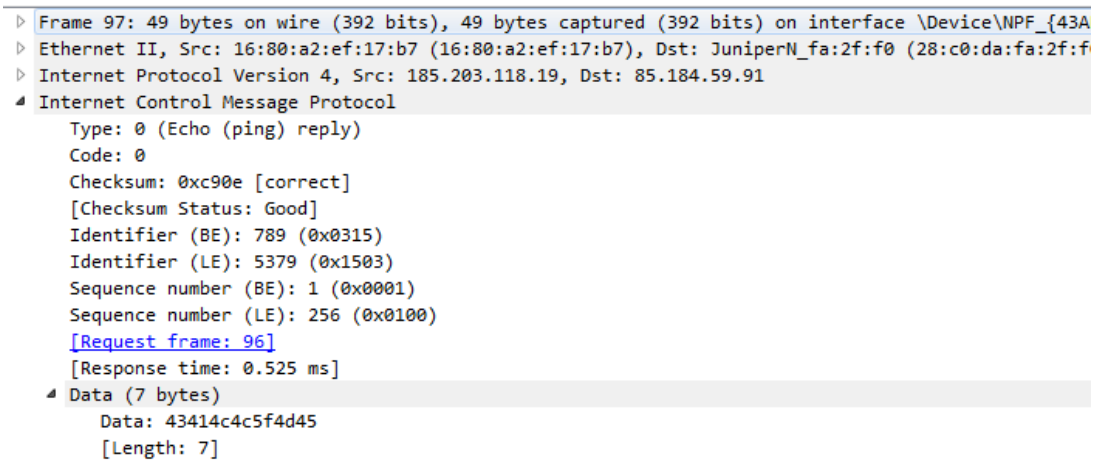


Figure 5.29. Wireshark " checksum packet status" reply

Experiment 4. Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experimental steps and doing the necessary analysis and correctness checks, it can be seen that the transmission was 100% correct without any obstacles or loss of data. Table 5.12 shows the relevant variables for this experiment.

Firewall	Yes
data is encrypted	*
Data before encryption	*
Data after encryption	*
Number of operations in the network	98
Number of packets for the ICMP process	2
RTT	.2 ms
network congestion>50	Yes

Table 5.6. Experiment No. 4 Results

5.2.5 Experiment No. 5

Experiment No. 5 is an external experiment to a **receiver outside the network with a juniper type firewall**. We will send a message of (**1362**) letters to the external receiver, and the results will be studied in this part.

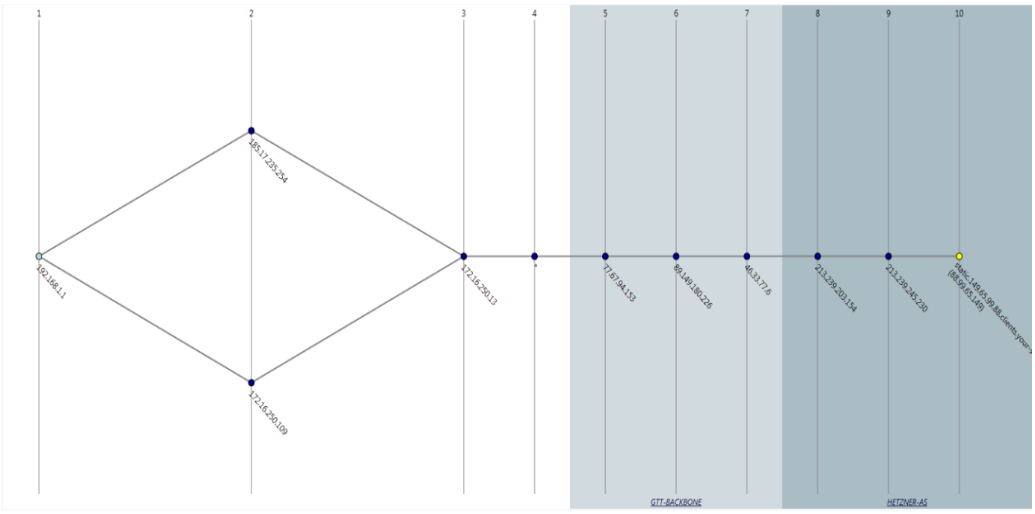


Figure 5.30. Path data flowchart

- The message that consists of (1362) letters were sent without "encoding" after being divided into parts through the experimental work environment.
- According to the following path, the packet passed through (8) global routers until it reached its final destination, as shown in Figure 5.31 & Table 5.13.

Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Duration	Payload Bytes Sent	Total Bytes Sent	Payload Bytes Received	Total Bytes Received	Missed Bytes	Packets Received
2019-11-15 21:48:18 Z	CXL8GI3KdmnB3hZwLf	192.168.1.116	3	88.99.65.149	3	icmp	80.05	1272	1608	1362	1390	0	1
2019-11-15 21:48:36 Z	C34IKiqUcmzw4v7h9	192.168.1.116	8	88.99.65.149	0	icmp	0.08	2724	2780	0		0	1
2019-11-15 21:48:37 Z	C15QIo4hNjjF4Olsl1	192.168.1.116	3	88.99.65.149	2	icmp	0	1096	1152	0	0	0	0

Hop	IP Address	Hostname	Time (ms)	Country	Status
1	192.168.1.1	?	1	Unassigned or assigned to IANA.org	11:0:The hop limit expired in transit
2	185.17.235.254	ADSL-185.17.235.254.mada.ps	75	PALESTINIAN TERRITORY	11:0:The hop limit expired in transit
3	77.67.93.9	ae0-165.cr3-fra2.ip4.gtt.net	73	GERMANY	11:0:The hop limit expired in transit
4	89.149.180.45	et-0-0-53.cr11-fra2.ip4.gtt.net	73	GERMANY	11:0:The hop limit expired in transit
5	46.33.77.6	?	82	GERMANY	11:0:The hop limit expired in transit
6	213.239.245.217	core21.fsn1.hetzner.com	89	GERMANY	11:0:The hop limit expired in transit
7	213.239.245.230	ex9k1.dc1.fsn1.hetzner.com	118	GERMANY	11:0:The hop limit expired in transit

Table 5.13. The table shows the data traffic and the addresses of the global entities that these packets pass-through

Map and traffic data packets are divided into three local lines, global Internet zones, and the host data center server for Test 5, as shown in Figure 5.32.

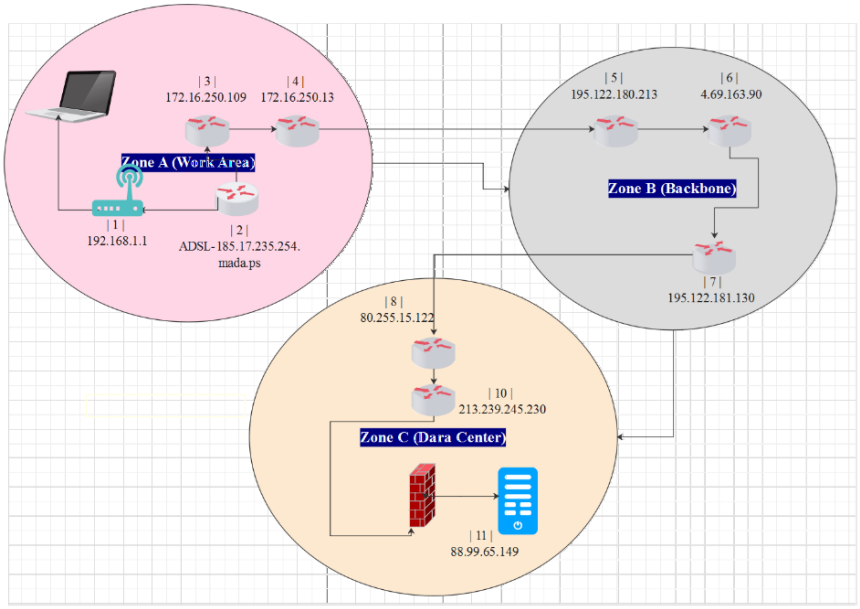


Figure 5.31. Local line, global Internet zones, and the host data center server

During sending the unencrypted packet, operations running on the network were captured inside the recipient's machine. We registered 85 network operations, and we noted congestion and overhead processing inside the queues, as shown in Figure 5.33. The figure shows the processes according to their relevant clarifications while capturing the packet.

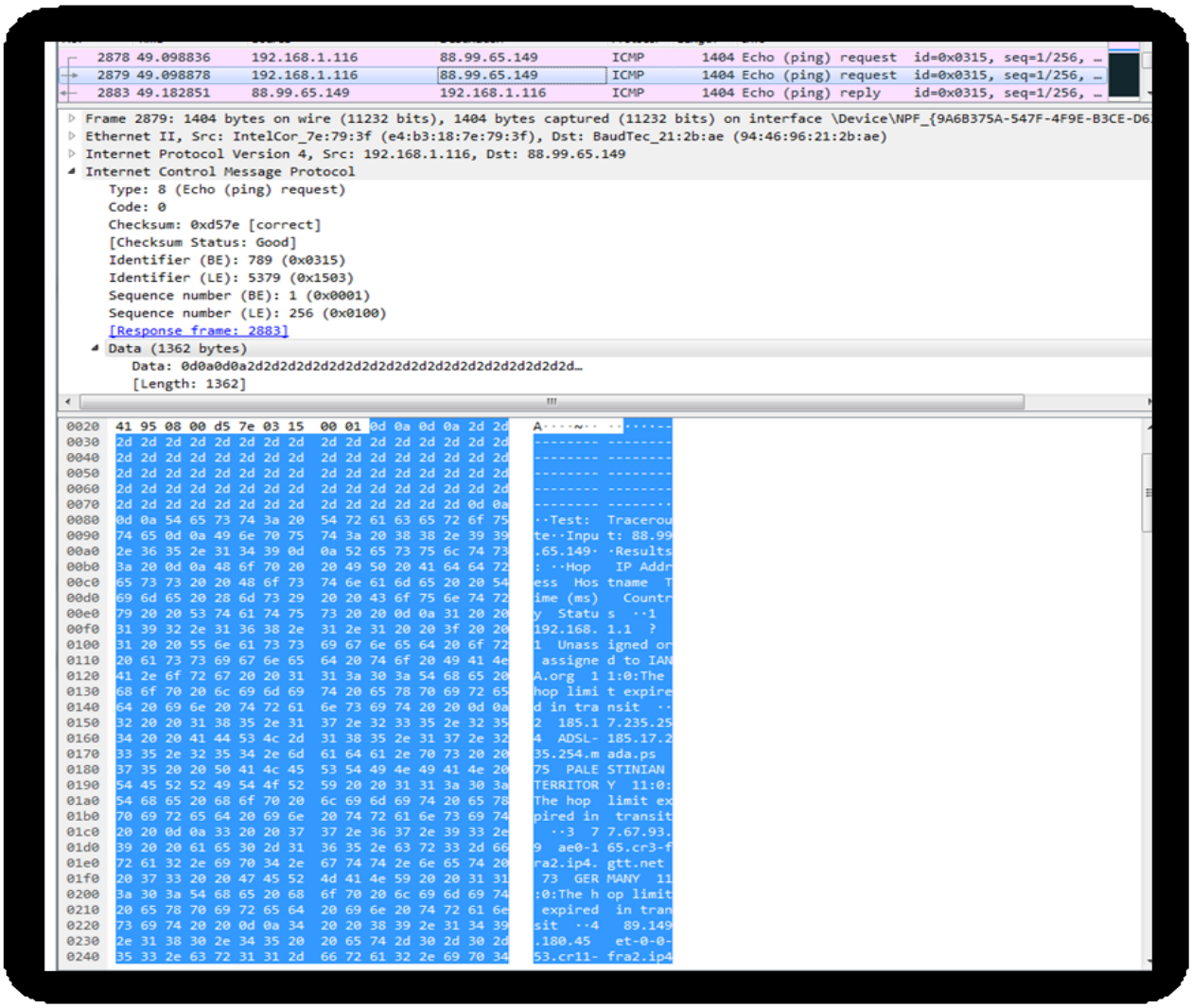


Figure 5.32. Operations during packet capture

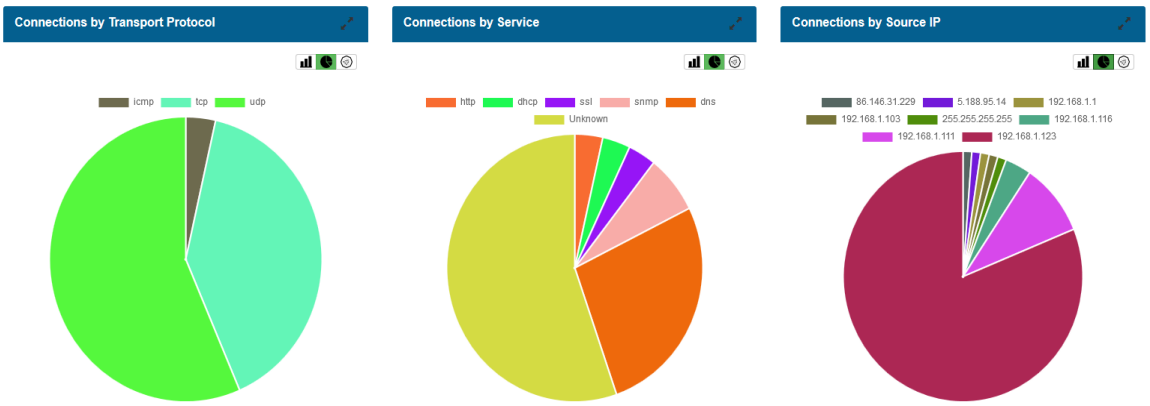


Figure 5.33. Distribution processes, service ports, and IP destination

The processes were categorized and filtered, but only ICMP-related processing was selected and had a connection id of (CXL8GI3KdmnB3hZwLf). We filtered and extracted six fields

Therefore, we have adopted the variables of Table 5.14 as follows: injecting 1362 characters into the ICMP packet, and the packet arrived as required and was 100% correct, without any loss of data. As shown in Figure 5.35, the time-lapse between sending and receiving was 83.9 MS.

Here we see the Wireshark capturing of the packet during the experiment; Firstly, the destination of the packets my computer sends (192.168.1.116) to (88.99.65.149). Second, we send 2 ICMP requests to the receiver, and we have a response with one ICMP reply; a total of 3 packets were sent

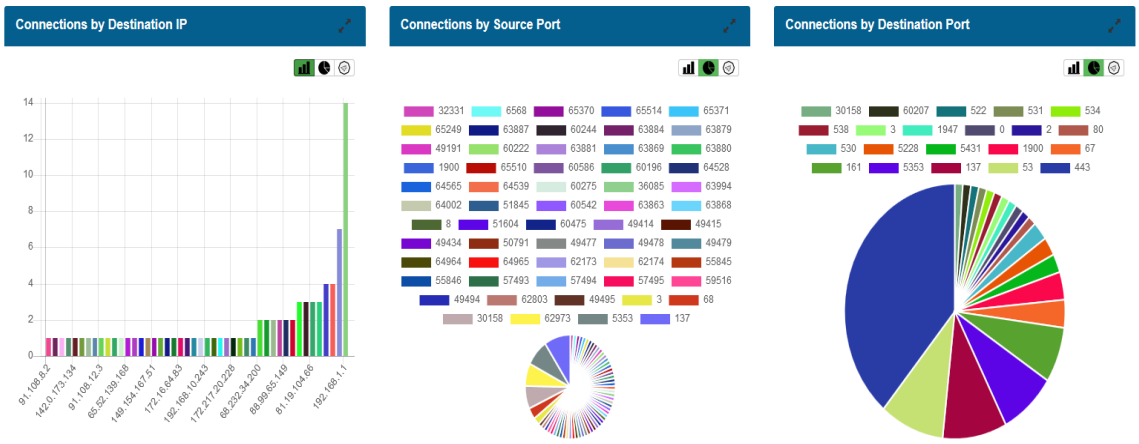


Figure 5.34. The packets after transmission and data injection and received.

In Figures 5.37 and 5.38, we verify that the packet arrived correctly.

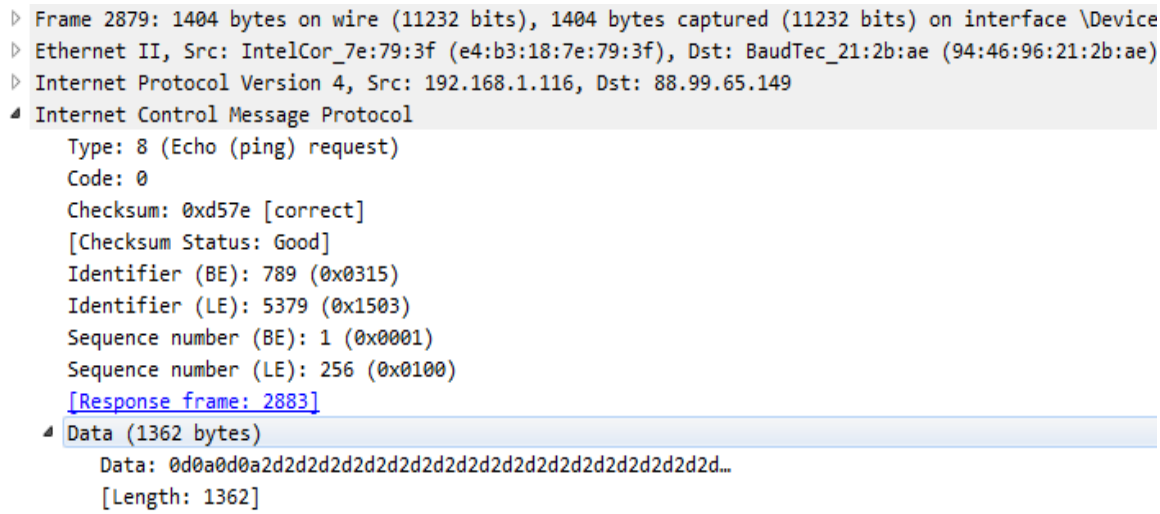


Figure 5.35. Picture of the Wireshark network monitoring software system

2879 49.098878 192.168.1.116 88.99.65.149 ICMP 1404 Echo (ping) request id=0x0315, seq=1/256, ttl=128 (reply in 2883)

2883 49.182851 88.99.65.149 192.168.1.116 ICMP 1404 Echo (ping) reply

Experiment 5. Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experiment's steps and doing the necessary analysis of the packet, and verifying the correct access to it, experiment No. 5 did not lose any of the packets. The transmission was 100% without any obstacles difficulties. Table 5.15 shows the relevant variables for this experiment.

Firewall	Yes
Data is encrypted	*
Data before encryption	*
Data after encryption	*
Number of operations in the network	85
Number of packets for the ICMP process	3
RTT	80.6 ms
Network congestion>50	Yes

Table 5.15. Results for experiment No. 5

5.2.6 Experiment No. 6

Experiment NO. 6 is a local experiment with a Kaspersky Endpoint Security v11 firewall. We want to send the message “0569747410” as an ASCII string to the local receiver.

The message will be sent without encoding after being divided into parts through the experimental work environment. During the process of sending the unencrypted packet, network operations were captured inside the recipient's machine. There were (85) operations with some congestion and

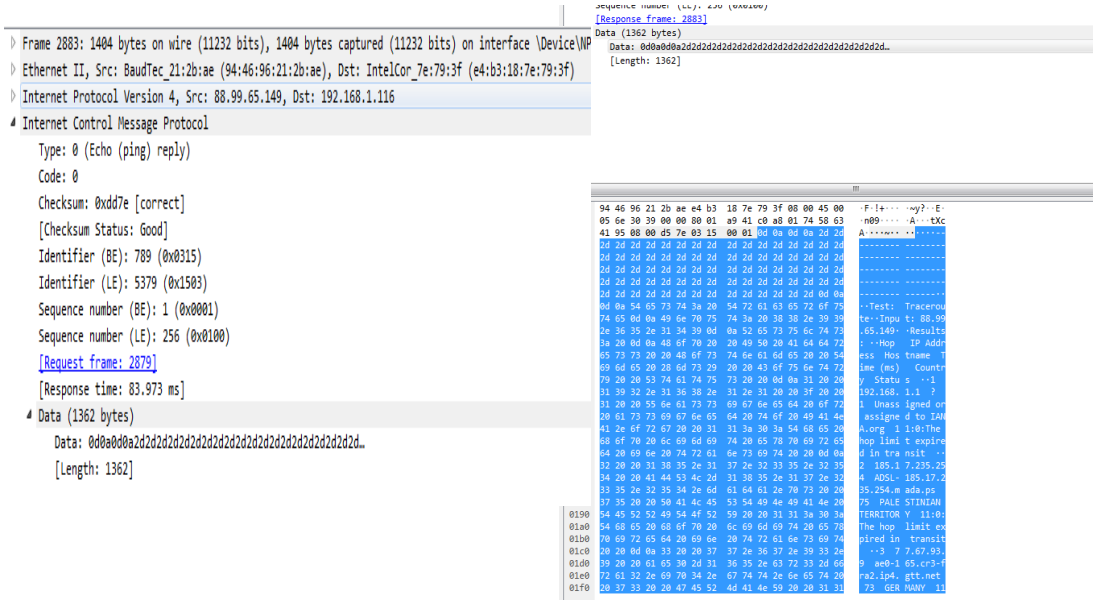


Figure 5.37. Wireshark " checksum packet status” reply

overhead processing due to a firewall. Figures 5.39 and 5.40 show the relevant processes according to their clarification.

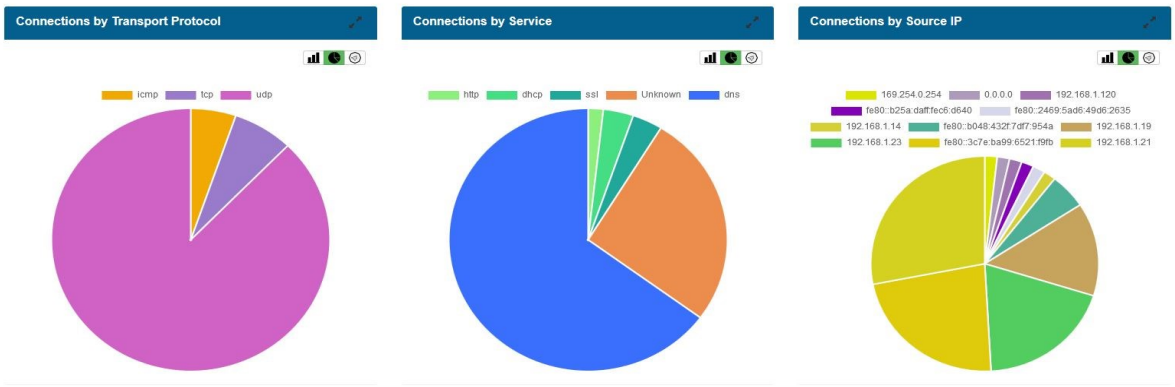


Figure 5.39. Operations during packet capture

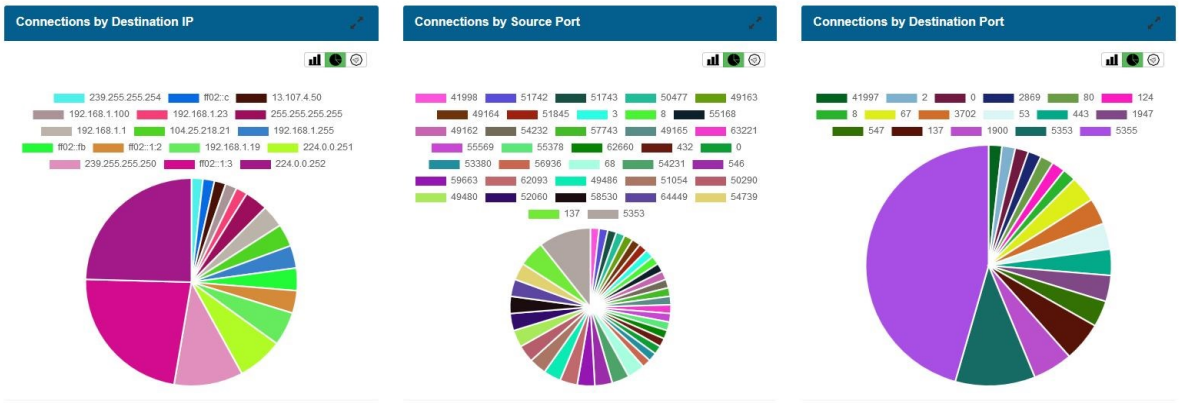


Figure 5.38. Distribution processes, service ports, and IP destination

The processes were categorized and filtered, and only the ICMP process was selected and had a connection id (CtQkUuwOLeDjbeEUd).

We filtered and extracted six fields, and they are shown in Table 5.16.

Timestamp	Connection ID	Sender IP	Target IP	Transp	Duratic	Payloa	Total B	Payloa	Total B	Missed	Packets	Packets Receiv
2019-11-17 14:47:51 Z	CtQkUuwOLeDjbeEUd	192.168.1.23	192.168.1.19	icmp	34.68	20	76	20	76	0	2	2

Table 5.7. Experimental data Test 6

A quick analysis of these variables follows: we injected a string of 10 digits into the ICMP packet, and the packet arrived as required, correct and without loss of data. The elapsed time (34.68 ms) was significant due to an active local firewall that captures and checks every packet.

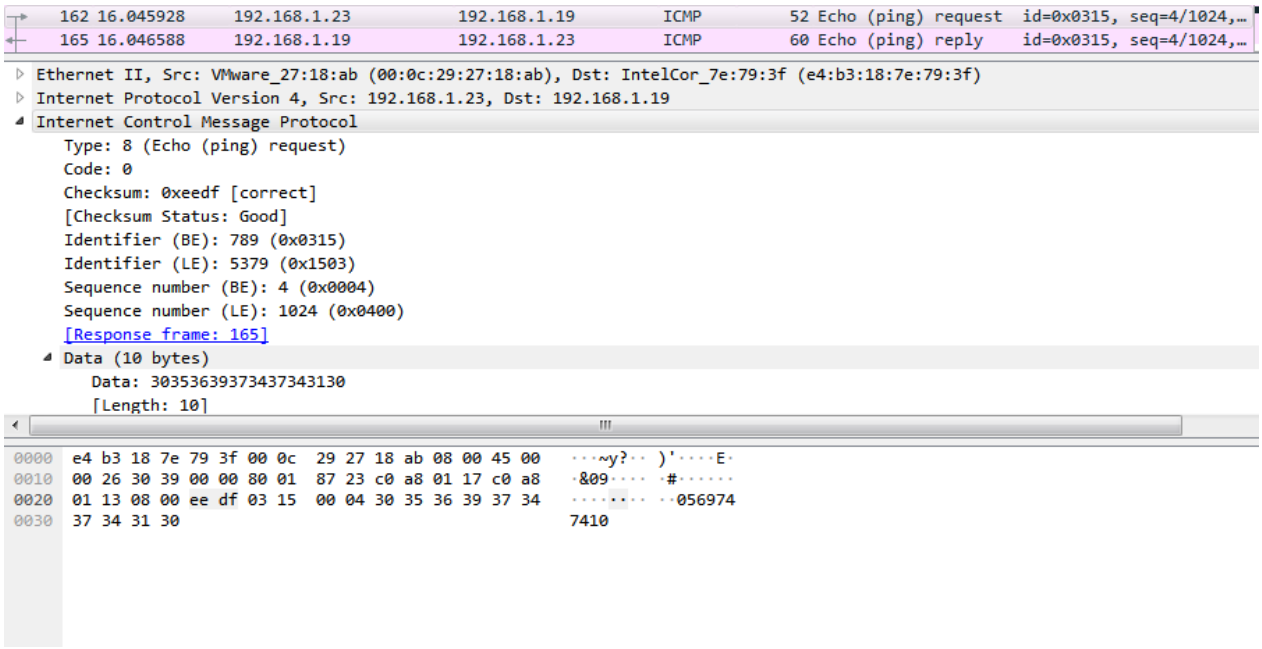


Figure 5.40. The packets after transmission and data injection

Figure 5.42 shows a snapshot of the Wireshark capturing of the packet. Here we can observe the following: Firstly, the destination of the packets my computer sends 192.168.1.23 to 192.168.1.19.

Secondly, we sent one ICMP request to the receiver, and we got one ICMP reply; a total of two

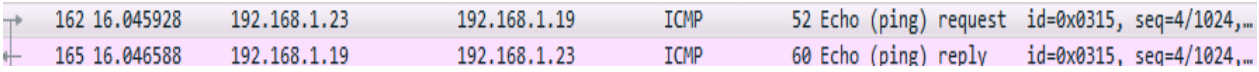


Figure 5.41. Picture of the Wireshark network monitoring software system

packets were sent and received.

Next, we want to verify that the packet arrived correctly (Figures 5.43 and 5.44).

162 16.045928 192.168.1.23 192.168.1.19 ICMP 52 Echo (ping) request id=0x0315,

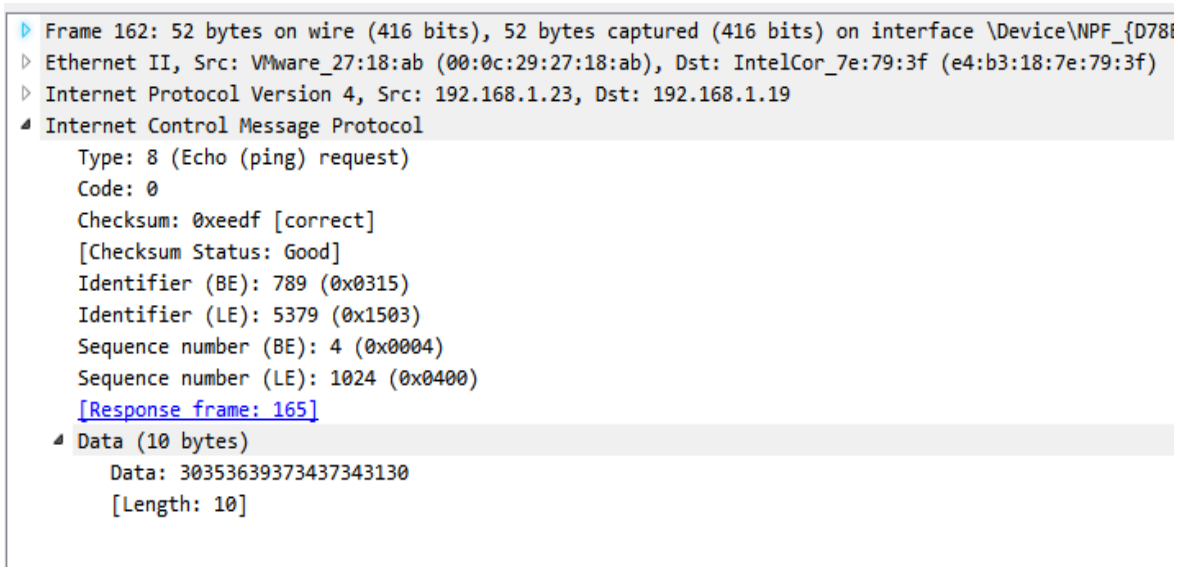


Figure 5.42. Wireshark “checksum packet status" request

seq=4/1024, ttl=128 (reply in 165)

165 16.046588 192.168.1.19 192.168.1.23 ICMP 60 Echo (ping) reply id=0x0315, seq=4/1024, ttl=128 (request in 162)

Experiment NO.6 Summary:

- Correctly inject data into the packet
- Correctly sending data inside the packet
- Receive properly without losing data

After performing the experimental steps and doing the necessary analysis of the packet, and verifying the correct access to it, experiment No. 6 did not lose any of the packets. The transmission was 100% without any obstacles or difficulties. Table 5.17 shows the relevant variables for this experiment.

Firewall	**Yes**
Data is encrypted	*
Data before encryption	*
Data after encryption	*
Number of operations in the network	**58**
Number of packets for the ICMP process	**2**
RTT	**34.68 ms**
Network congestion>50	**Yes**

Table 5.17 Results for experiment No. 6

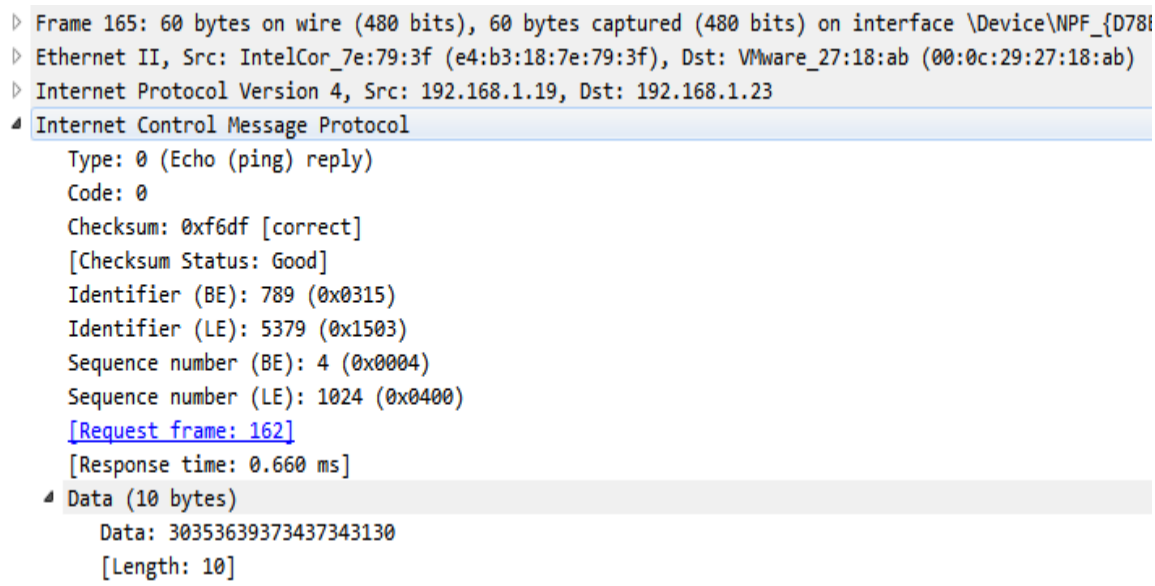


Figure 5.43. Wireshark " checksum packet status" reply

5.2. Experiments conclusions

Through the application of the proposed concealment techniques to conceal identified information using the hidden channel technology, and through the results obtained in the experiments, the following conclusions can be reached:

Table 5.8.Summary of the experimental results

Experiment	Firewall	data is encrypted	Number of operations	number of packets for the ICMP process	RTT in ms	Congestion
1	NO	YES	4	64	34.95	NO
2	NO	YES	6	16	6.21	NO
3	NO	YES	6	16	7.22	NO
4	YES	NO	95	2	0.2	YES
5	YES	NO	85	3	80.6	YES
6	YES	NO	58	2	34.68	YES

1. There is an invisible connection that is detected between the sender and the receiver using hidden channels, according to experiment No. 5 and experiment No. 4. These two experiments were performed in the presence of a firewall and outside the local network with a difference in the number of packets and the number of operations occupied by the receiver and the presence of network congestion as the packets arrived correctly and adequately.

2. Experiment No. 6 was carried out between two devices on a local network with the Kaspersky endpoint security system 11. The message and data arrived without encryption correctly in the presence of operations inside the system and congestion in processing.

3. In experiments 1, 2, and 3, we had a data encryption and packet splitting process, and all the packets were correctly received and were properly assembled without any packet or data loss.

4. Injections and transmissions in Experiments 1, 2, and 3 were done similar to the injections off-grid experiments of 4, 5, and 6. The only difference was in processing time due to the local network's data passing via global routers and incurring reasonable delay due to the transmission distance and within the expected elapsed times.

5. In each packet, one letter (or digit) was sent according to the settings of experiments 1, Experiment 2, and Experiment 3. The difference was in the time taken to send the packet to the receiver side. The difference could be due to the differences in the number of operations and of the packets.

6. The time of transmission and arrival of the experiments differed inside and outside the network by a small amount.

7. The ICMP protocol provides masking spaces to pass up to 1472 characters per packet.

IP header	**ICMP header**	**ICMP payload size**	**MTU (1500)**
20 bytes	8 bytes	1472 bytes (maximum)	20 + 8 + 1472 = 1500

** Number of entries inside the header may not exceed the size of 1472 for the packet to be sent correctly, reliably, and properly*

8. According to Experiments 4, Experiment 5, and 6, the firewall did not detect any packet as being harmful or had errors in the header.

9. When the size of the packet was maximum (Experiment 5), it was successfully sent. However, it took a relatively long time to reach its destination. Specifically, when sending the entire space that can be used within the packet, the packet arrives correctly but after a relatively long time compared to the rest of the experiments. This is due mainly to the processes within the operating system.

10. Experiments differed in the transmission time factor, as experiments in an internal and external network environment amounted to parts of a second.

11. When sending in the form of distributing letters to packets, it takes more time than one packet to send.

12. The encryption process will increase the time spent in the transmission process. Still, it will maintain the transmission's confidentiality and privacy in the event of any interference or capture from a third party.

5.3 Solution Systems

The primary goal of this study is to collect evidence by having the ability to view packets correctly and legally. Here in forensic network security, for example, the network focuses on detecting and monitoring network security concerns, such as piracy activities. This can be accomplished by irradiating the searchlight when breaking into patterns to investigate internet crimes, ensuring no leaks or injections at any of the headers and no incorrect behavior.

The experimental test results indicate that using the monitoring system might require specific modifications to any network structure to be implemented. This is to capture the packets that may threaten information security or to indicate if there is a possibility of exploitation.

Some security measures and some network structures do not allow the packet's interception or capture packets due to a policy and rules in the firewalls or other secondary devices.

In some network architecture devices, the packet capture service helps identify the network problem, and the captured packets are stored in the buffer or exported to other parties when registering the network packets. The goal is likely to know how web applications interact with different network applications, network workflow, and endpoint scanning. This is required for network protection and the detection of any abnormal behavior.

In this area, HP has provided some software solutions, but only provided them to discover errors and network behavior and not discover malware or injection of headers or packets. This technology did not provide packet analysis from within its header. It was presented in its network devices that provide such service or additional service to which the scale of performance in Bandwidth, Latency (Round Trip Time), and Packet Loss is attached.

The search for a solution to this problem and the new type of cyber-attack contributing to data leakage has already begun. It has many symptoms for network and system managers. In addition to tracking this type of problem, we will suggest a solution that might represent a future study, its applicability, cost-effectiveness, and efficiency.

5.4. A summary of the service within HP network devices

It has become apparent to us that HP company already provides the service, but it must be manually started from within the device attached to the network, such as the switch or the router. This falls within the framework of the technician or person's authority so that he has access to the settings or is authorized to enter to operate this feature. You need high access privileges so that you can have access to certain settings as well. Not every technician inside the network department has the right to enter the system of network devices or their settings and he does not analyze any procedure. He only performs the task assigned to him. He does not intercept or disclose and does not save more than the employee's session time. A technician on duty can export the data to another place, but this is not an ideal solution. Another problem is that it only saves a maximum of 10 packets, and there are challenges and difficulties in working with this system. Recent development in the system include:

"A capture or display filter contains a keyword string or multiple keyword strings that are connected by operators. Keywords include the following types:

- Qualifiers—Fixed keyword strings. For example, you must use the IP qualifier to specify the IPv4 protocol.
- Variables—Values supplied by users in the required format. For example, you can set an IP address to 2.2.2.2 or any other valid values. A variable must be modified by one or multiple

qualifiers. For example, to capture any packets sent from the host at 2.2.2.2, use the filter src host 2.2.2.2.

- Operators include the following types:
- Logical operators—perform logical operations, such as the AND operation. • Arithmetic operators—perform arithmetic operations, such as the ADD operation.
- Relational operators—indicate the relationship between keyword strings. For example, the = operator indicates equality.

5.6 A summary of the service inside Cisco network devices

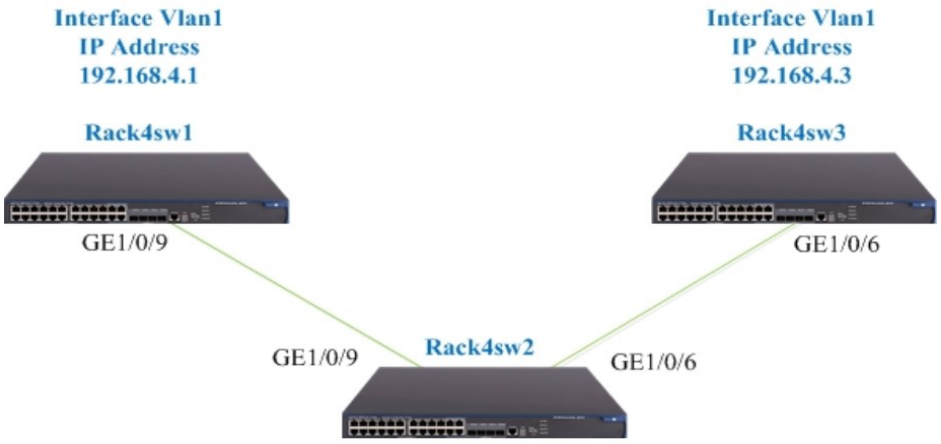


Figure 5.44. Packet capture switching

Cisco system has similar drawbacks to those of the HP system [9], including:

- In releases earlier than Cisco IOS Release 15.0(1) M, the buffer size was limited to 512K.

[9] **Web-site: https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-embedded-packet-capture/116045-productconfig-epc-00.html visit: 05/12/2019 00:37**

- In releases earlier than Cisco IOS Release 15.0(1) M, the captured packet size was limited to 1024 bytes.
- The packet buffer is stored in DRAM and will not persist through reloads.
- The capture configuration is not stored in NVRAM and will not persist through reloads.
- The capture point can be defined to capture in the cef or process switching paths.
- The capture point can be defined to capture only on an interface or globally.
- When the capture buffer is exported in PCAP format, L2 information (ethernet encapsulation) is not preserved.
- See Best Practices for searching Commands to obtain more information on the commands used in this section

5.7 Prototype Solution Model

HP and Cisco developed and implemented specific solutions and systems in order to overcome this study problem. However, the current research suggests a more efficient mechanism to deal with data leakage. The suggested system (Figures 5.45 and 5,46) consists of practical solutions, methods, tools, and network design. It requires detailed engineering of the entire network (careful rebuilding of all network activities for users and hackers, protocols and applications, network performance analysis and troubleshooting, data mining, data leak detection, data breach investigation, quick response to cases and mitigation, troubleshooting, and tracking). A packet capture program is an essential element of the proposed system. The program inserts itself into a network stack to extract copies of and store frames before sending them to a final device. This is repeated for the received packet data. Following the OSI model, contestants like *TCPDUMP* upon posting will move the bits on the data link.

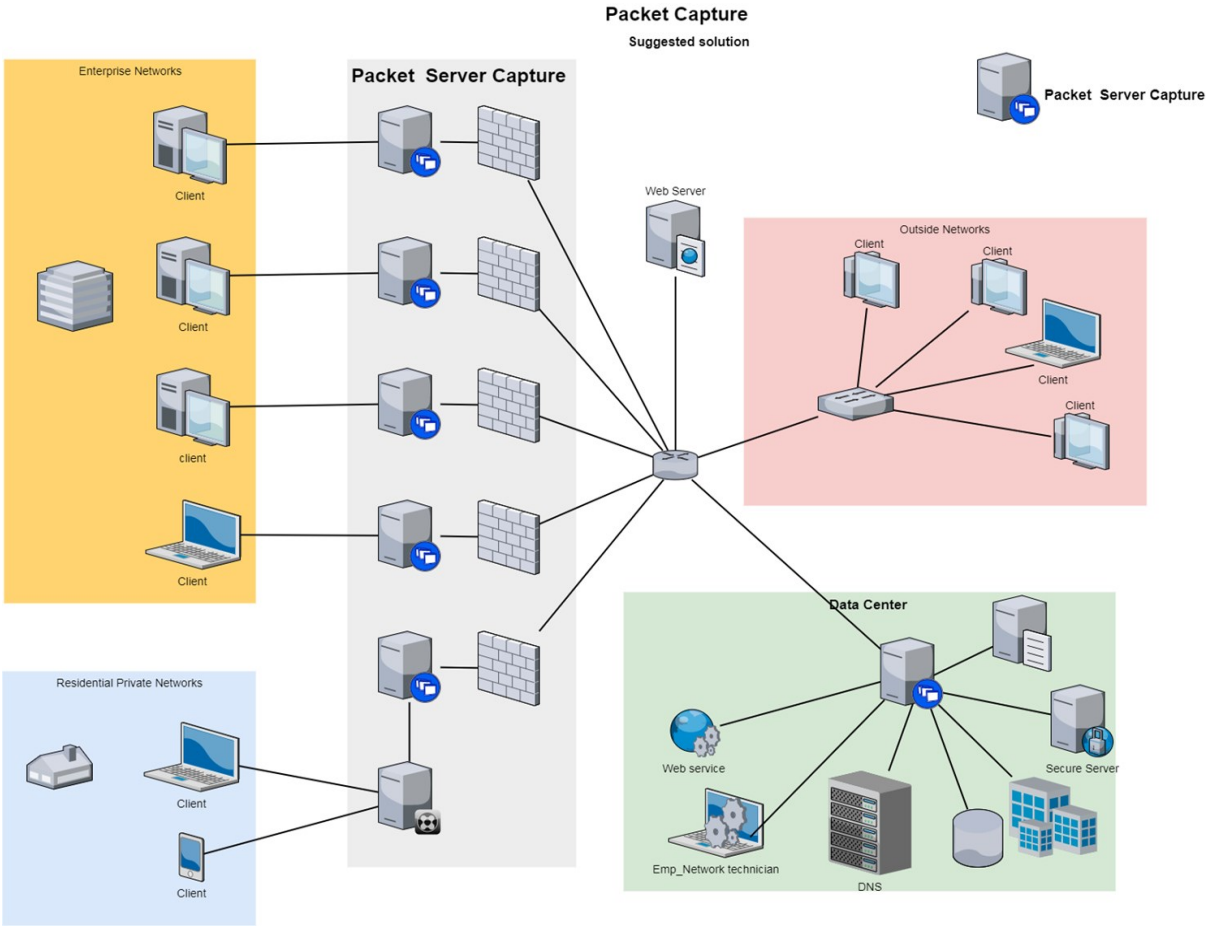


Figure 5.45. Packet capture Suggested solution

Using the free packet capture software and Packet Sniffing, Wireshark is a free and open-source software for network eavesdropping and packet analysis. It is used for network error detection, network analysis as well as educational purposes. The project was initially named Ethereal and was renamed in May 2006 due to brand selection.

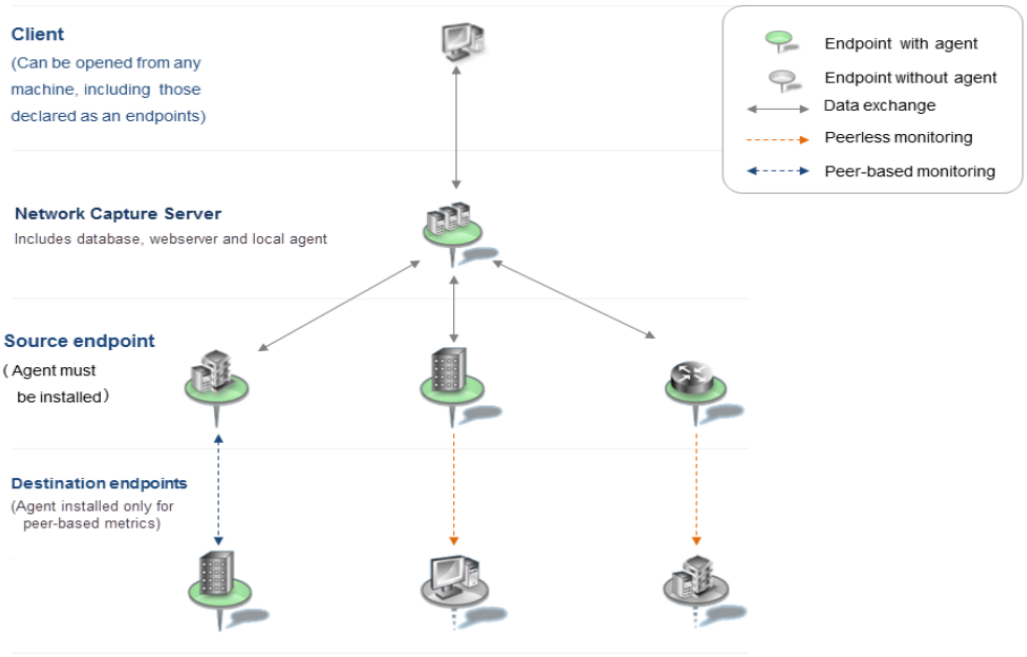


Figure 5.46. Suggested solution 2

It is also one of the most well-known tools used in hacking distributions. Its main task is to eavesdrop on tools and see data passing through the network, whether written, visible, or audible.

So, it is always better to have a packet capture appliance (full) installed in your network to give details that can be used to reconstruct web sessions for thorough investigations of current and past network activities and provide a real-time view of the network.

The Wireshark multi-platform program works on Linux, Windows, Mac OS, BSD, Solaris, and some other Unix-like operating systems, and Wireshark filters are of two types:

- Capture Filters: These are filters that you specify to be recorded. These filters are specified before the recording/capture pickup operation process.

- Display Filters: These are filters used to search inside data/information recorded or captured by Wireshark. They can be used while Wireshark logs packets.

The first type of filtering is used to reduce the recorded packets' size and thus not obtain a record of huge packets. The second is more powerful but more involved simultaneously and is used to search for specific data or information within the registered packets.

This type of filter's syntax is the same as used by programs that depend on the Lip cap library. For such programs as *TCPDUMP*, the filter must be set before running and registering/picking up packets by Wireshark. Any modification of these filters must restart the registration/capture process from the beginning. We cannot modify them during the registration process, as in the Display type filters.

The second type is a filter that is used to search within packets registered with Capture Filter. The capabilities available to search through this type are very high and broad, exceeding what the first type of filters can do. Any amendment to these filters does not require restarting the registration/capture process from the beginning. We can add, delete, and modify them during the registration process.

There are many advantages to this simple solution:

- Low cost
- Scan hundreds of LAN protocols, and add more new protocols to be scanned continuously.
- A full analysis of the internal network.
- Files from this program can be exported to PCAP or CSV or even to plain text.
- Live Data can be read through networked platforms

[PCAP value] is constrained by some factors, depending on the size of the networks captured, their speed, number, and areas, and volume of data that the system will collect and store, which are all clear challenges.

we need to organize and try to solve these challenges by adjusting the internal work policy to suit the size of any problem or danger surrounding the network system and by organizing the policy of operating the proposed system manually when exploring errors or problems session.

Chapter 6

Conclusions and Future Work

Chapter 6 : Conclusions and future work

In this chapter, we present the packets' analysis results and the nature of the study problem that was reached. We also describe study samples as well as some discussion of the study questions and their answers.

The science of discovering errors and treating them within the network depends on monitoring these networks' behavior, extracting results from their behavior, predicting the existence of a problem, and attempting to find a solution based on the diagnosis.

Digital forensics can be described as the science of identifying, extracting, and maintaining computer records, cookies, cache, metadata, Internet searches, and any other legally accepted evidence. These methods and techniques can be used to solve cyber-crimes committed using the internet-connected infrastructure.

Internal control has witnessed a continuous development in its concepts, starting from looking at its importance in protecting assets from theft and misuse and ensuring the validity of information and the detection of violations and deficiencies to monitoring packets within the network to maintain the confidentiality of data and processing it then and adequately discover errors within networks.

We have addressed this problem during our study and exploited other issues that need to be addressed and studied.

6.1 Study Conclusions and Analysis

Based on the above, in this section, we summarize the study in response to the questions and hypotheses mentioned before.

The results are consistent with our research hypotheses, as the ability to inject data into the ICMP header may contribute to data leakage outside the network without the possibility of tracking this leak, in addition to the hypothesis that there is no supervision or follow-up within the information technology policies to address this problem. Most importantly, we provided practical and real proof of a new type of cyber-attacks using the ICMP protocol.

Result #1:

After a technical study and real experimentation and analysis, we have identified a new type of cyber-attacks. Please see Figure 6.1, which explains this new type of attack.

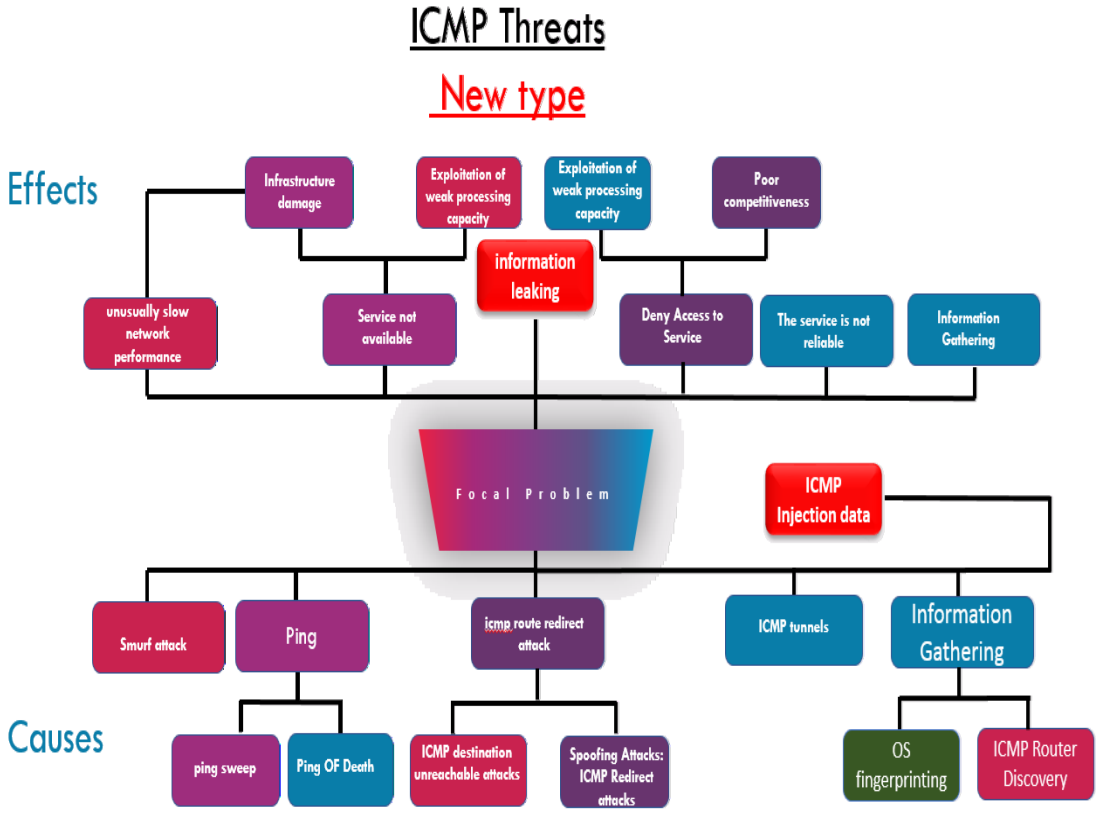


Figure 6.1. ICMP Threats & New type

Result #2:

The network protection system procedures do not analyze the packet's behavior to ensure no significant traffic impurities. There are several standard security checks and tools inside the network protocols and systems. Still, in our study, we have revealed that these systems and/or checks simply did not succeed in revealing or catching our injections.

One of the most critical protection functions is monitoring the packets, filtering them, and treating them. Hence we have a problem with bypassing the firewall. Indeed it succeeded if there was a necessary condition on the packet that it should be within a specific size by dividing the information to be leaked out of the network to its standard size and pass as seen in experiments 1 to 3. The data arrived correctly without errors and without being detected by the firewall.

Result #3:

One question appeared in network security circles: How much power should we grant to the network administrators to give them access to the network tools and devices, to capture malicious packets (i.e., injections and leakage, etc.)? It turns out that this exploitation by giving high authorities to some employees can lead to excessive use of these privileges. Some employees inside the network may contribute to the harm or gain some knowledge that he/she is not entitled to know or access illegally.

There is a problem in companies and authorities' internal policies, the extent of responsibilities related to repair follow-up, and the identification of problems within the network layers. There is a problem since these policies and procedures must be updated regularly. Also, there are rules and procedures to regulate technicians' entrance into the Data Centers of these companies (e.g., a request for physical access to the data center). These visits must be organized and regulated: what is the nature of the visit? What is the problem that this technician is dealing with? Must explain everything that he/she will be doing inside the data center, etc. Our study shows that such a scenario is possible, and data leakage can also happen in such cases of physical access to the Data Center.

Result #4:

There is no impact on the information and network security methodology if the ICMP network protocol is used to leak or inject confidential data abroad. The CIA triangle (Confidentiality, Integrity, and Availability) will affect one of the essential criteria: data confidentiality and data center privacy and information. There is a negative impact on protocol exploitation regarding breach of confidentiality, integrity, and information standards availability.

Result #5:

There is no significant difference between the typical size of the original packet and the exploited packet, bearing in mind that it will remain within the agreed terms in building protection systems, network structure, treatment method, building headers, and network packet formation and protocols.

Result #6:

There is no impact on the system's efficiency to handle other operations to exploit this type of security risk. The packet can be sent even when it contains the maximum of its swabs, as there is work inside the network tools and equipment while maintaining a condition that it does not exceed 1500 MTU. After the testing procedures, we had the credibility and validity of the packet in terms of conducting mathematical calculations related to the header to ensure that it was perfect.

Our experiments have proven that the packet is valid through checksum procedures as seen by the transmitter and that network equipment and devices are a prerequisite for this packet to be considered valid. It is known that any modification to the packet that gives error values in the checksum will take into account the network attachments of the switches, firewalls, or charging equipment. This packet is considered invalid or has been tampered with. According to some devices, it is possible to ignore the packet or to indicate that it has not been verified, or it has impurities. Such devices can neglect or restrict these doubtful messages.

Result #7:

The researcher's suggestion is a prototype of techniques that limit abnormal behavior to exploit the ICMP protocol. During the research, there are techniques, tools, equipment, and programs that limit abnormal behavior to exploit the ICMP protocol, such as firewalls, identifiers, IPS, anti-virus applications, firewalls applications, and network analysis tools. We have previously mentioned that there are tools to deal with such problems, but they are limited due to space challenges and the ability to manipulate devices inside the network equipment.

6.2 Future work

In the future work that will be based on relevant studies and the results of our study, where we will work to build a system for tracking and following us and the summary of this study: "An Internet worm is a type of malicious software (malware) that self-replicates and distributes copies of itself to its network. The Internet worm attacks different destination victims by using an Internet protocol that supports three main protocols, TCP, UDP, and ICMP, regarding transport and control protocol. This research designed an algorithm that focused on ICMP protocol scanning attack only. When the algorithm focuses on the different protocols, the detection will be difficult. There are several detections for internet worm. Most of these detections depend on General Protocol Scanning Detection (GPSD). Our technique focused on internet worm behavior when the research used ICMP scanning and failure connection for ICMP protocol. However, the GPSD technique focuses on general failure connection received by different protocols. When the research compared GPSD and proposed algorithm, the research found the proposed algorithm h faster detection when the worm used ICMP scanning than a technique that depends on GPSD."

The answer to our study questions will be counted as a starting point for future work as well.

References

Book:

[1] Bharti, Vishal and Snigdh, Itu (2008), "Practical Development and Deployment of Covert Communication in IPV4", Birla Institute of Technology, India
[2] Church, Roberto Gómez Cárdenas, and Ryosuke Watanabe,2004, Data Hiding in Identification and Offset IP Fields, Enrique
[3] David Llamas (2004), Covert Channel Analysis and Detection using Reverse Proxy Servers, School of Computing, Napier University, EH10 5DT, Scotland, UK
[4] Dr. Naseer Ali Husieen, Dr. Mohammad Maher Rasheed (2015). Detection Algorithm for Internet Worms Scanning that Used Internet Control Message Protocol
[5] Eric Mitchell, Channel SE, East US, and Federal, F5 Networks
[6] F. Baker, Ed, 1995, Requirements for IP Version 4 Routers.
[7] Frozen, Behrouz A. (2007). Data Communications and Networking (Fourth Ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 978-0-07-296775-3.
[8] Forouzan, Behrouz A. (2007). Data Communications and Networking (Fourth Ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 978-0-07-296775-3.
[9] HP Network Capture, (2015), Software Version: 7.11 page 321,311
[10] HPE 5500 EI Switch Series - How to use the Packet Capture Utility, Article Number mmr_sf-EN_US000005595
[11] Postel, J. (September 1981). Internet Control Message Protocol
[12]
[13] Sebastian Zander, Grenville Armitage, Philip Branch (2007). Covert channels in the IP time to live field. Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Melbourne, Australia.
[14] WJ Buchanan and Dr. Llamas 2004, Covert channel analysis and detection using reverse proxy servers.
[15] Zander, Sebastian, Armitage, Grenville, and Branch, Philip (2006), "Covert Channels in the IP Time to Live, "Swinburne University of Technology Australia

Website:

[16] https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model model "The OSI Model's Seven Layers Defined and Functions Explained." Microsoft Support. Retrieved 2014-12-28
[17] https://www.a10networks.com/blog/5-most-famous-ddos-attacks/
[18] https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-embedded-packet-capture/116045-productconfig-epc-00.html visit: 05/12/2019 00:37
[19] https://www.iana.org/assignments/-parameters/icmp-parameters.xhtml
[20] https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml#icmp-parameters-types
[21] https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml#icmp-parameters-types